UNIVERSIDADE DE LISBOA
Faculdade de Ciências
Departamento de Informática

EPIDEMIC MARKETPLACE: REPOSITÓRIO E WEB SERVICES

Carla Patrícia Freitas Sousa

MESTRADO EM ENGENHARIA INFORMÁTICA
Especialização em Sistemas de Informação

2012

UNIVERSIDADE DE LISBOA
Faculdade de Ciências
Departamento de Informática

EPIDEMIC MARKETPLACE: REPOSITÓRIO E WEB SERVICES

Carla Patrícia Freitas Sousa

PROJECTO

Projecto orientado pelo Prof. Doutor Mário Jorge Gaspar da Silva
e co-orientado pelo Prof. Doutora Maria Dulce Pedroso Domingos

MESTRADO EM ENGENHARIA INFORMÁTICA
Especialização em Sistemas de Informação

2012

Agradecimentos

Após todo este tempo de trabalho e partilha não poderia deixar de nomear aqueles que durante este percurso me apoiaram, incentivaram e ajudaram a trilhar este caminho.

As minhas primeiras palavras de agradecimento vão para a minha família, especialmente aos meus pais e padrinhos de baptismo, pelo apoio prestado, pela compreensão e claro por estarem sempre a torcer por mim.

Ao Professor Mário J. Silva, pela confiança que depositou em mim, pela força e motivação que me transmitiu. Ao Professor agradeço tudo aquilo que me ensinou; as bases que me deu para o desenvolvimento do meu saber, o espírito de equipa que aprendi a respeitar; a confiança no desempenho do cargo de Administração do Grupo XLDB e a liberdade de acção que me permitiu. Tudo me ajudou a crescer e a tornar-me numa cidadã mais activa, empenhada e apta para o mundo do trabalho.

À Professora Dulce Domingos pela sua total disponibilidade e simpatia para co-orientar esta tese e ainda por me ter transmitido o interesse por estas matérias.

À Professora Ana Paula Afonso por me ter acolhido e orientado no meu primeiro projecto (GREASE).

Às Professoras Teresa Chambel, Ana Paula Cláudio, Isabel Nunes e ao Professor Paulo Urbano pela partilha das suas experiências que contribuíram para a minha decisão em ingressar neste mundo que sempre me fascinou mas que por algum motivo fui adiando.

Aos Professores Fabrício Silva e Francisco Couto pela sua compreensão e ”Savoir faire”e apoio constante sempre com orientações claras, objectivas, dirigidas ”ao alvo”e concretas, sem subjectividades, permitiram a sua realização, isto é, tornar verdadeiro algo a que desde sempre me propus e que, no fundo, apesar de ser uma etapa, é como uma ”maratona”no mundo avassalador de tantas sinergias e energias que nos cercam e circundam.

A todos os meus colegas, e especialmente ao Nuno Cardoso, Hugo Ferreira, Luís Filipe, João Zamite, João Ferreira e Juliana Duque, com quem partilhei experiências, conhecimentos e que tão fundamentais foram na troca de ideias e saberes.

Ao Tiago Sousa pelo apoio, incentivo incondicional, pela sua bondade e por tudo o que representa para mim.

Ao Sérgio Serafim e Bruno Tavares pelos projectos em que tive o privilégio de os ter como colegas e que proporcionaram-me grandes momentos de loucura e diversão como

que uma pitada de sal condimentando um maior e melhor empenho nas tarefas a realizar.

Não posso deixar de mencionar também, todos os meus amigos que sempre me acompanharam e conviveram comigo nos últimos anos. Em especial, aos amigos do Colégio A Minha Escola e aos do Atletismo, pela força, companheirismo e momentos de descontracção e alegre convívio que me ajudaram a superar momentos menos bons.

Por último, mas não em último, aos meus colegas do Voluntariado por toda a partilha e entrega que me transmitiram.

E, provavelmente os agradecimentos não ficariam por aqui. E aos que, por alguma razão, foram obliterados ou ”esquecidos”, não no sentido profundo da palavra, porém, como traição de memória, aqui ficam os meus agradecimentos em jeito quase de desculpa.

Contudo, a vida é feita de amizades e os amigos são-no para sempre.

Dedicatória.

Para os meus pais que tanto me apoiaram e à restante familia que sempre se prontificou a ajudar-me

Resumo

É cada vez mais recorrente o uso da *Internet* para armazenamento de dados científicos, dando primazia a uma melhor organização dos mesmos de forma estruturada e simples, permitindo a cooperação entre a comunidade científica e contribuindo para a melhoria da sua qualidade ao longo do tempo. O projeto aqui apresentado teve como objectivo a concepção e desenvolvimento de um repositório digital com *web services* para acesso e gestão do seu conteúdo. O trabalho decorreu no âmbito do projecto Europeu *Epiwork*, que visa a criação de um sistema de previsão, detecção e simulação de surtos epidémicos. Foi realizado com o objectivo de responder a necessidades sentidas com o desenvolvimento e utilização de uma primeira versão.

O *software* do repositório foi construído sobre uma plataforma *Fedora Commons* com um sistema *XACML* de autenticação e autorização. O repositório pode ser acedido interactivamente através de um sistema de gestão de conteúdos *Drupal*. Este utiliza as funcionalidades que os *web services* desenvolvidos oferecem para aceder e manipular os recursos epidemiológicos do repositório. Os *web services* fazem a mediação entre os clientes e o repositório através de uma interface *RESTful*, que transfere, além dos conteúdos, metadados nos formatos *OAI-ORE* e *OAI-PMH*.

Palavras-chave: Repositório Digital, Metadados, Sistema de Gestão de Conteúdos, *Web Services*

Abstract

The use of the Internet for scientific data storage is becoming prevalent, enabling their structured organization, the cooperation among the scientific community and their improvement over time. The objective of this project was the design and implementation of a digital repository with web services for epidemic data management. The project took place within the Epiwork project, which has the goal of developing a system for the prediction, detection and simulation of epidemic outbreaks. The development is intended to answer the needs identified with the development and use of an initial version.

The software of the repository was built on the Fedora Commons framework, with an XACML user authentication and authorization system. The repository has an interactive interface based on the Drupal Content Management System. It uses the developed web services for access to the epidemic resources in the repository. The web services implement a RESTful interface exchanging contents and structured meta-data in the OAI-ORE and OAI-PMH formats.

Keywords: Digital Repository, Metadata, Content Management System, *Web Services*

Conteúdo

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

"Preparar o ambiente adequado para a recepção de um novo Sistema de Informação é um dos factores críticos do sucesso da sua implementação. A mudança dos processos que lhe estão associados é sempre difícil e, para ter sucesso, deverá começar a ser preparada desde a fase de concepção do projecto de desenvolvimento, acautelando os factores chave para a mudança."(Catarina Torres [34])

Hoje em dia estamos rodeados de informação. Para a difusão desta informação é necessário que exista um suporte para o seu armazenamento para, por exemplo, ser usada num estudo ou pesquisa. Para tornar isto possível e de acordo com Torres é preciso conceber novos sistemas de informação [34].

Nos últimos anos um enorme fluxo de dados quantitativos sociais, demográficos e comportamentais ficaram disponíveis na internet. Estes dados podem ser utilizados por modelos matemáticos e estatísticos para providenciar mais rapidamente uma melhor localização e capacidade de detecção além de melhorar os tradicionais sistemas de vigilância de doenças, que são muitas vezes fundamentais para o seu estudo e controlo.

Os dados utilizados nos sistemas de vigilância, quanto mais exactos melhor, devem ser modelados de forma a se conseguir uma melhor compreensão da propagação de doenças e avaliação do seu impacto na saúde pública. Embora estes dados, contenham a informação precisa e necessária, muitas vezes não se encontram nem centralizados nem organizados para serem facilmente encontrados e partilhados entre cientistas e os diversos profissionais de saúde.

Recentemente as redes sociais como o *Facebook* ou o *Twitter* aparecem, com a mediatização que os caracteriza, como fortes fontes de obtenção de dados epidemiológicos. As pessoas são muitas vezes levadas a exporem dados do foro epidémico na sua rede social, à espera que todos os seus amigos fiquem a conhecer o seu estado de saúde. Este tipo de acção ajuda a perceber a dispersão de uma doença na comunidade, fornecendo a base para qualquer tipo de estudo.

É sabido que a ocorrência de surtos epidémicos, normalmente à escala mundial, gera

um maior volume de dados na internet explicando assim a importância que os sistemas de vigilância têm.

No entanto, este grande volume de dados, deixa em aberto um problema, o armazenamento e categorização dos mesmos. É necessário garantir que estes dados são correctamente guardados, geridos e disponibilizados para a comunidade científica num sistema único.

É neste contexto que surge o projecto EPIWORK. Este projecto propõe um esforço multidisciplinar de investigação com o intuito de desenvolver uma plataforma de ferramentas e conhecimento necessárias ao desenho de infra-estruturas de previsão epidemiológica por epidemiologistas e outros investigadores da área da saúde – o *Epidemic Marketplace* (EM). É um projecto Europeu e conta com a participação de 12 organizações de investigadores entre os quais se encontra o LASIGE (Large-Scale Informatics Systems Laboratory), que desenvolve projectos na área da informática biomédica e na área da gestão de informação, entre outras.

O *Epidemic Marketplace* visa melhorar o acesso aos dados epidemiológicos e permitir uma maior colaboração entre epidemiologistas e outros profissionais de saúde.

Os dados são geridos e disponibilizados aos utilizadores através de um repositório digital, de acordo com regras de acesso bem definidas.

O *Epidemic Marketplace* tem os seguintes módulos (figura 1.1):

1. Repositório: Guarda conjuntos de dados (*datasets*) epidemiológicos e ontologias que caracterizam a semântica deste conjunto de dados. O repositório guarda estes *datasets* e os seus metadados. Este pode ser acedido em `http://v2.epimarketplace.net/`.

2. Colecção de dados (*Data Collection*): Realiza a recolha de mensagens que contêm as palavras-chave das doenças e localização geográfica, a partir das redes sociais na internet, nomeadamente do *Twitter*. A API do *Data Collection* pode ser acedida em `http://www.epimarketplace.net/documentation/documentation.html`.

3. *Web Services*: Tem por objectivo a integração automática dos dados epidemiológicos provenientes de diferentes fontes. Os *Web Services* têm a possibilidade de recolher automaticamente informação das aplicações, processar os dados e armazená-los no repositório. A API dos *Web service*s pode ser acedido em `http://v2.epimarketplace.net/developers_corner`.

O meu trabalho consistiu no desenvolvimento da segunda versão do Repositório e dos *Web Services*.

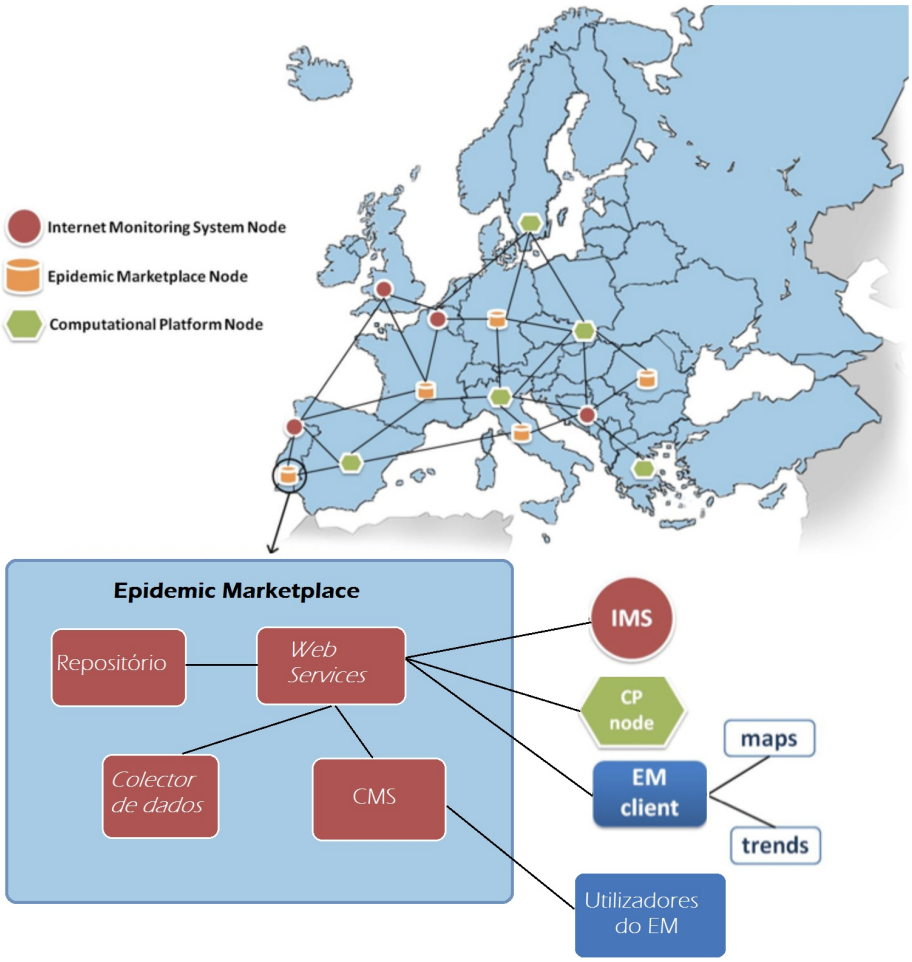

Figura 1.1: Representação da *Epidemic Marketplace*

1.1 Objectivos

O desafio deste projecto residiu principalmente na realização de um repositório perceptível ao utilizador e conceber uns *Web Services* com melhor desempenho e com mais funcionalidades que a versão anterior. Portanto, os objectivos para este projecto foram:

- Disponibilizar a versão 1.0 do *Epidemic Marketplace* para o público em geral;
- Desenvolvimento da versão 2.0 do *Epidemic Marketplace*, incluindo o repositório em *Fedora Commons*, a *interface* gráfica para acesso a esse repositório em *Drupal* e os *Web Services* em *Python* para acesso a partir de aplicações;
- Implementação de uma *interface RESTful* nos *Web Services*, que obedeça aos principios da arquitectura *REST* e que permita às aplicações aceder aos dados de forma elegante e rápida;

- Implementação do protocolo *OAI-PMH*, responsável pela consulta e partilha de metadados na *Internet*;
- Implementação de um sistema de pesquisa no repositório, que visa a busca de informação sobre o novo modelo de dados do Epidemic Marketpace;
- Disponibilização da versão 2.0 do *Epidemic Marketplace* para o público em geral.

1.2 Metodologia

"Try and fail, but not fail on try"(Stephen Kaggwa)

Este projecto envolveu a implementação de duas componentes, o repositório onde é guardada a informação relativa à epidemologia e a componente *Web Services* que realiza a gestão dos dados, nesse repositório, para as aplicações. Para isso, o desenvolvimento deste projecto envolveu fases independentes e complementares, para que as funcionalidades dos *Web Services* e a *interface* do repositório fossem adequadas à comunidade de epidemologia e não só.

As fases de desenvolvimento deste projecto foram:

- Análise: Nesta fase foi recolhido o *feedback* da utilização da versão 1 da plataforma e realizado um novo levantamento de requisitos consoante as necessidades sentidas na utilização de todo o sistema. Entenda-se todo o sistema como sendo o repositório, a sua *interface* e *web services*. Este *feedback* foi essencialmente a visualização dos *logs* da plataforma e a quantidade de objectos digitais que foram inseridos na plataforma. O levantamento de requisitos incidiu com a vinda de um membro do projecto *Epiwork* a Portugal. Com este membro foi possível perceber a sua dificuldade na utilização do *EM*. Procedeu-se à realização da técnica "Think Aloud", onde o utilizador era levado a realizar algumas funcionalidades do sistema e no decorrer desse processo, o utilizador, ia referindo em voz alta o seu raciocínio. Com isto foi possível reunir alguma informação que deu a base de desenvolvimento da nova versão do *Epidemic Marketplace*;
- Pesquisa: Foi necessário reformular quer o repositório quer os *web services*. Para isso foram realizadas pesquisas para a procura de melhor o *software*, que realizasse em menos tempo as tarefas do sistema. A pesquisa foi feita com base em novas tecnologias e *open source*. Por intermédio de vários testes, de usabilidade e de desempenho, realizados à antiga versão do *EM* conseguiu-se perceber que era necessário reformular os serviços *web*. Ocorreu a dúvida entre duas linguagens, *Perl*

ou *Python* mas acabei por decidir por utilizar uma linguagem que já conhecesse o *Python*.

- Desenho: Foi modelado todo o sistema com base no *feedback* e requisitos para uma melhor percepção e divisão de tarefas ao longo do tempo que foi definido para a conclusão do projecto. Esta fase foi essencialmente restruturada devido ao facto de utilizadores da plataforma já se terem queixado do *design* da antiga versão, através de reuniões internacionais.

- Implementação: Percebendo o problema encontrado na fase de análise e tendo uma lista de *software* que foi encontrado na fase de pesquisa, é realizada uma escolha tendo em conta a curva de aprendizagem do novo *software*, o tempo definido para esse fim e a actualidade do *software*. Esta pesquisa recaiu para o *software* em que eu sentisse mais afinidade e facilidade de desenvolvimento.

- Avaliação: Foi avaliado o desempenho e possíveis problemas, quer na *interface* quer nos *web services*. Estes testes foram concebidos com base no *feedback* dos utilizadores e a realização de testes de stress a todo o sistema. Caso tivesse sido necessário modificar alguma componente ter-se-ia realizado uma nova pesquisa para cumprir eficientemente a necessidade sentida.

Na figura 1.2 pode ver-se o diagrama com as diferentes fases de desenvolvimento do projecto.

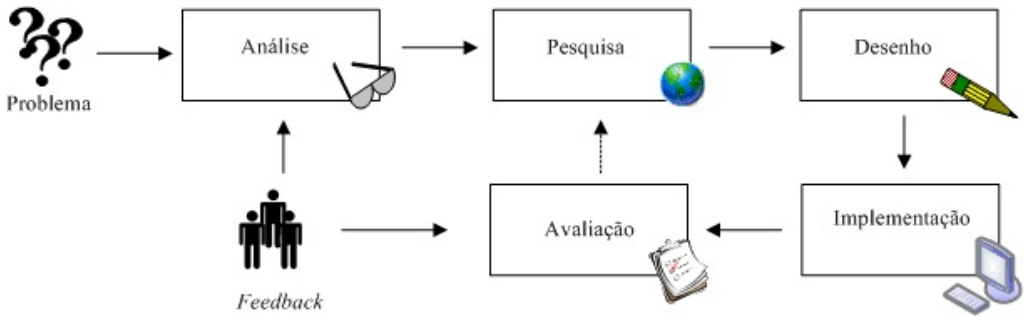

Figura 1.2: Diagrama das fases de desenvolvimento do projecto

1.3 Resultados

A realização do projecto foi delineado em quatro grandes objectivos (secção 1.1), que posteriormente tiveram que ser subdivididos em tarefas mais simples para uma melhor coordenação de tempo de realização.

Para desenvolvimento da versão 2 do *Epidemic Marketplace*, foi necessário primeiro proceder a uma formatação de dois servidores e posteriormente à colocação de um sistema de virtualização, *Xen*. Esta tarefa exigiu uma curva de aprendizagem considerada elevada, uma vez que eu não possuía qualquer conhecimento nesta matéria. No entanto reconheço que foi uma grande aposta ao introduzir uma estrutura deste tipo no *Epiwork*, tendo daí colhido a vantagem de não ter de partilhar os recursos das minhas máquinas virtuais com outros utilizadores.

Após o obstáculo da virtualização do sistema, prossegui com o desenvolvimento dos serviços *web* em *Python*. Já tinha conhecimento da linguagem, contudo, o que me foi mais difícil de implementar foi o sistema *web* em *Python* que culminou num atraso de duas semanas. Deparei-me com a situação de este tipo de sistemas serem desconhecidos para a grande maioria dos internautas, não conseguindo obter ajuda com exemplos ou *foruns* de ajuda. A solução encontrada foi por tentativa/erro e por uma pesquisa muito específica em *HTTP*, nomeadamente em *Apache*.

O facto de ter definido à partida um tempo para realizar pesquisas sobre como desenvolver os serviços *web*, permitiu-me recuperar o tempo perdido. O facto de estar inserida no projecto *Epiwork* desde o seu inicio, proporcionou-me o conhecimento total de todas as ferramentas/abordagens utilizadas no *Epidemic Marketplace*. Com isso foi possível ter conhecimento sobre o protocolo *OAI-PMH* ou a estrutura *OAI-ORE*, que aparentemente era o que eu tinha mais receio de demorar a implementar. No entanto, a implementação dos *web services* correram na sua maioria sem qualquer tipo de dificuldade.

Consegui reformular e implementar os novos serviços *web* em menos tempo do que o previa, no entanto, o serviço *web* que me deu mais trabalho foi o *search*. Este foi, sem qualquer dúvida, o obstáculo maior que encontrei ao longo da implementação do meu projecto. A grande dificuldade foi a compreensão de como o *software solr* funcionava e como este se ligava ao *Fedora Commons*. Entre tentativas falhadas, leituras exaustivas da documentação e pesquisas muito aprofundadas no *Google* consegui concretizar um sistema de pesquisa que incide sobre o novo modelo de metadados do *EM*.

A implementação do ambiente gráfico em *Drupal* não exigiu uma curva de aprendizagem muito grande.

Após a realização deste projecto e dos vários obstáculos encontrados, posso afirmar que sinto um orgulho muito grande em o projecto *Epiwork* ter serviços *web* em *Python* e um sistema de pesquisa sobre o novo modelo de metadados do *Epidemic Marketplace*.

1.4 Estrutura do Documento

Este documento está organizado em cinco capítulos de acordo com a seguinte descrição:

- Capítulo 2 - Pesquisa de todo o *software* para conceber os dois módulos do *Epidemic Marketplace*, repositório e *web services*;

- Capítulo 3 – Descreve os requisitos do *Epidemic Marketplace*;
- Capítulo 4 – Descreve a implementação dos dois módulos do *Epidemic Marketplace*;
- Capítulo 5 - Conclusões com trabalho futuro.

Capítulo 2

Trabalho Relacionado

Este capítulo é fundamental para perceber quais as ferramentas que construíram o *Epidemic Marketplace* e o porquê dessa escolha. A primeira versão do *Epidemic Marketplace* ajudou a definir a linha de pesquisa não deixando, contudo, de ser *software open-source*.

A pesquisa realizada, para a concretização da nova versão do *Epidemic Marketplace*, foi ”partida”em várias componentes. Neste caso era preciso um repositório e um *front-end* para a visualização dos seus dados. Para a implementação do módulo *web services* teria que ir perceber dos vários serviços *web* existentes qual seria o melhor para o caso do *Epidemic Marketplace*. Por fim, perceber qual a melhor forma de garantir segurança dos dados e autenticação dos utilizadores.

2.1 Bibliotecas Digitais Vs Bibliotecas Tradicionais

> *”A memória das máquinas impõe-se pela sua grande estabilidade, enquanto materialidade e é sustentada pelo efeito de transparência produzido pela reprodução e condensação dos registos. Por outro lado, ela reproduz aquilo que já se tornou a memória do Homem, algo parecido ao tipo de memória que representa o livro, mas combinada, no entanto, com uma facilidade de evocação até então desconhecida”(Lucas,1998)*

As bibliotecas têm vindo a alterar o suporte utilizado nos seus acervos após o Homem ter descoberto a escrita. No inicio estas eram compostas por minerais escritos através de hieróglifos, passando depois para pergaminhos e papiros e mais tarde o papel.

As bibliotecas foram, durante muito tempo, locais reservados às altas classes sociais e por isso as informações eram tuteladas pela nobreza, clero e magistratura. Os livros eram presos às prateleiras para não serem emprestados e também para um maior controlo da informação [38].

A partir do século XVI, as bibliotecas sofreram profundas alterações e adquiriram características laicas, democráticas, especializadas e sociais, no entanto só, no século XVIII, a informação começou a chegar às outras camadas das Sociedade [38].

No século XIX, dado o volume de memória já existente implementou-se o recurso a fichas, para um registo de todo o acervo da biblioteca [38].

Já no século XX, a introdução dos computadores provoca uma verdadeira revolução da memória, com a utilização da memória electrónica, ilimitada e dependente de técnicas de armazenamento [38].

Foi no início dos anos 90, que o Laboratório Nacional de Los Alamos, nos Estados Unidos, construiu e implantou o repositório digital *arXiv* (primórdios das bibliotecas digitais), com o objectivo de experimentar uma alternativa para a comunicação científica, possibilitando agilizar a publicação dos resultados das pesquisas e facilitar o acesso à comunidade. Desta forma, os pesquisadores podiam depositar os seus resultados de pesquisa, quer na forma de *papers*, quer na forma de relatórios técnicos num repositório digital de livre acesso.

A biblioteca tradicional tem, assim, vindo a perder terreno, relativamente à biblioteca digital, que muitos consideram ser, mesmo, a *Internet*. Até há poucos anos era normal uma pessoa usufruir de uma biblioteca tradicional para explorar algum tema que fosse proposto para um trabalho escolar, uma questão que surgiu no momento, ou simplesmente requisitar um livro para a sua leitura diária.

Com a evolução da *Internet* as bibliotecas digitais foram ganhando força em relação às tradicionais proporcionando aos utilizadores novas funcionalidades que eram, até então, inexistentes. Este aumento de uso culminou no reforço do conceito de biblioteca.

Uma biblioteca digital pode ser considerada como um espaço virtual onde o conteúdo é totalmente digitalizado e armazenado num repositório em diversos formatos. Associado a este repositório de dados existe um sistema de informação com as funcionalidades necessárias à manipulação do repositório de acordo com as necessidades dos utilizadores [46].

Estas bibliotecas foram criadas com o objectivo de dar à comunidade acesso a mais informação proveniente da *Internet*.

É fácil de perceber que existem várias vantagens no uso de bibliotecas digitais em detrimento das bibliotecas tradicionais.

Mais à frente, apresento uma listagem de algumas características que se destacam nas bibliotecas digitais.

Deter-me-ei, sobre o último ponto da lista, visto ser um assunto de extrema importância para a compreensão do meu trabalho.

- Permite o acesso remoto do utilizador, por meio de um computador com recurso à *Internet*, ou seja, o utilizador pode aceder ao repositório de qualquer parte do mundo à hora que lhe apetecer. Uma das grandes desvantagens das bibliotecas tradicionais

são o facto de terem um horário muito restrito e muitas vezes encontram-se muito distantes umas das outras, o que dificulta, por vezes, a pesquisa considerando o factor tempo como essencial.

- Permite a utilização do mesmo documento por várias pessoas em simultâneo. No caso das bibliotecas tradicionais pode não ser possível, por exemplo, o mesmo livro estar disponível para vários utilizadores por não haver exemplares suficientes. Assim a partilha da informação é uma realidade e os resultados pesquisatórios compensarem o que é deveras importante.

- Permite a utilização de diversos suportes de registo de informação, tais como texto, som e imagem. Neste caso a biblioteca tradicional pode não permitir este tipo de suporte.

- Não ocupa espaço físico. O utilizador pode, através de qualquer sistema de armazenamento de dados, guardar vários livros do seu interesse, o que permite uma maior gestao do seu tempo e uma comodidade acrescida.

- Contém um sistema inteligente ou especialista para ajudar na pesquisa de informação relevante. Este é, sem dúvida, um ponto importante. As bibliotecas tradicionais não dispõem de um sistema de pesquisa inteligente, ou seja, apenas é possível pesquisar sobre temas/assuntos e não sobre o conteúdo do documento propriamente dito.

A *World Wide Web (web)* é sem dúvida uma excelente fonte de dados.

A *web* foi criada com a visão de que seria um espaço onde a informação teria um significado bem definido, facilitando a cooperação e a comunicação entre as pessoas e os agentes de *software* [33].

Actualmente dado o acesso mais barato às tecnologias e *Internet*, qualquer pessoa consegue publicar informação, pessoal ou organizacional, contribuindo para uma expansão e desorganização dessa mesma informação.

Este crescimento da *web* fez com que a descoberta e recuperação de informação fosse penosa, devido maioritariamente a grande parte da informação disponibilizada ser entendida pelo ser humano o que torna dificil o processo de pesquisa e recuperação por mecanismos automáticos.

As bibliotecas tradicionais sempre nos habituaram ao uso de técnicas de organização para facilitar a busca de informação, como entre outras a diferenciação das estantes por tema/assunto.

As bibliotecas digitais, aparentemente, tornam-se mais fracas, no entanto, proporcionaram uma reviravolta ao mostrarem as suas capacidades de recuperação da informação relevante num aparente caos em que a informação se apresentava. Estas, as bibliotecas

digitais, utilizam os metadados para documentar e organizar, de forma estruturada a sua informação.

Os metadados são dados que descrevem outros dados, atributos e conteúdos de um documento original [24]. Estes tanto podem descrever imagens, textos, áudios, gráficos, etc. Por exemplo, podemos querer descrever uma imagem e o seu metadado poderia ser "É uma fotografia tirada em Lisboa. A pessoa que a tirou chamava-se Alice, que trabalhava na Faculdade de Ciências da Universidade de Lisboa".

Desde cedo, se percebeu que a utilização de mecanismos de pesquisa eficientes em base de dados e a escolha correcta dos metadados fez com que se melhorasse a razão entre o número de documentos relevantes retornados e o número total de documentos retornados isto numa pesquisa realizada pelo utilizador.

A descrição dos dados ainda se pode considerar muito pouco desenvolvida. No entanto, a definição de padrões de metadados tem vindo a ser utilizado pela comunidade para uma melhor comunicação entre os diferentes repositórios.

Os padrões de metadados facilitam a compreensão, integração e o uso partilhado de informações entre utilizadores. Estes padrões, uma vez estabelecidos implicam o compromisso entre utilizadores e provedores de informação (quem publica informação) que devem ambos aceitar, colaborar e usar as terminologias e as definições estabelecidas.

Dublin Core [10] é um formato padrão de metadados e é constituído por um conjunto de elementos descritores. O objectivo deste é identificar e definir um conjunto minimo de elementos capazes de descrever "objectos do tipo documento"(*Document Like Objects-DLOs*), numa ampla quantidade de recursos digitais. Desta forma o *DC* reúne uma solução que é comum à comunidade de bibliotecas digitais.

Um exemplo de uma descrição em *Dublin Core* e pegando no exemplo da fotografia, pode ser visualizado no excerto de código abaixo.

```
Title: Cheira a Lisboa
Subject: Paisagem sobre Lisboa
Description: Fotografia tirada com vista sobre a cidade de Lisboa
Language: Português
Creator: Alice
Type: Fotografia
Date: 07/07/2007
```

Desde cedo se percebeu que a implementação de uma biblioteca digital é algo complexa. No entanto, já existem ferramentas que nos ajudam a concretizar essa tarefa. A maioria destas ferramentas são *open source* e geralmente são desenvolvidas pelas Universidades e disponibilizadas livremente aos utilizadores [45]. Só assim foi possível concretizar um requisito que foi imposto ao *Epidemic Marketplace*. (Para mais detalhes ver o Capítulo 3 deste documento).

Tem sido referido ao longo deste capítulo a palavra Repositório. Um repositório é o local onde o conteúdo digital e os recursos são armazenados, podendo ser pesquisados e recuperados para uso posterior [25]. Este tem três funções principais:

- Garantir armazenamento e perservação dos recursos
- Garantir o acesso a esses recursos
- Garantir a recuperação dos recursos

O repositório suporta mecanismos de importação, exportação, identidade, armazenamento e recuperação de recursos digitais. Alguns exemplos de bibliotecas digitais serão aqui referenciadas, tendo como objectivo fazer uma breve apresentação da sua arquitectura e implementação. Dos exemplos apresentados consta aquele que foi seleccionado para o repositório.

2.1.1 *DSpace*

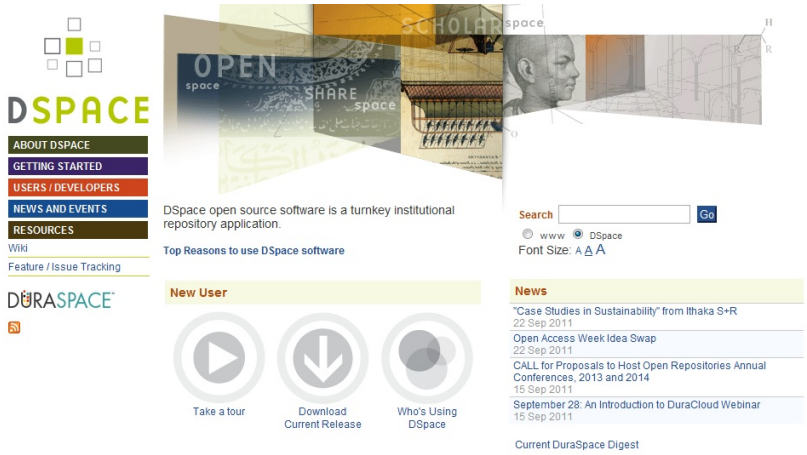

Figura 2.1: Página principal do *DSpace*

O *DSpace* é um *software* escolhido para ambientes académicos, sem fins lucrativos e para organizações comerciais. Este foi produzido pela *MIT Libraries e Hewlett-Packard (HP)* [8]. A figura 2.1 apresenta a página principal do *Dspace*.

É um *software* livre, fácil de instalar, é *out-of-the-box*, ou seja, não precisa de mais nenhum *software* para poder exercer as suas funções e é personalizado às necessidades do utilizador.

O *DSpace* foi desenvolvido na linguagem *Java*.

O modelo de informação no repositório é constituído por comunidades, colecções, *items* e *bitstreams* [9]. As comunidades são unidades de pesquisa da organização, enquanto que as colecções são grupos distintos de *items*. Já os *items* são vistos como objectos com conteúdo e, por fim, os *bitstreams* são vistos como ficheiros individuais.

Os dados são guardados no repositório em forma de objectos digitais, *items*, com um identificador único. Este identificador pode ser visto como o número do antigo BI e tem o nome de *handler*. Torna-se útil quando se deseja pesquisar dentro do repositório e permite a este objecto digital a sua longevidade e a sua preservação.

Outra das características em relação à informação contida no repositório é o tipo de *metadados*. *Dublin Core*[10] é o formato padrão dos *metadados* usados no *DSpace*, no entanto, segundo os seus criadores, "é possível adicionar ou alterar qualquer campo para costumizar a aplicação"[7].

O *DSpace* também oferece o suporte ao protocolo OAI-PMH [22]. Este protocolo permite a interoperabilidade entre outras bibliotecas digitais.

Sendo este protocolo primordial ao desenvolvimento do *Epidemic Marketplace* e tendo sido por mim utilizado na implementação do repositório, penso ser pertinente fazer uma breve abordagem, conquanto que sucinta.

A iniciativa OAI (*Open Archives Initiative*), criada em 2000, surgiu com o objectivo de minimizar a dificuldade que existia em partilhar os *metadados* entre diferentes repositórios. Vem da consequência de cada repositório implementar o seu próprio protocolo. É, neste contexto, que surge o protocolo OAI-PMH (*Open Archives Initiative – Protocol Of Metadata Harvest*), em que foi definido um conjunto de regras de comunicação nas quais todos os respositórios teriam que as seguir, caso pretendessem a partilha da sua informação. Desta forma é possível a transferência de dados entre dois repositórios digitais. O protocolo pode ser utilizado para ir buscar os metadados que estão no repositório, para posteriormente serem utilizados pelas aplicações externas. Na figura 2.2 é possível ver essa comunicação: a aplicação/cliente faz um pedido ao repositório e é-lhe devolvido um ficheiro que está de acordo com o protocolo OAI-PMH. Assim a aplicação/cliente, que fez o pedido, obtém os dados para depois serem utilizados por outras aplicações externas.

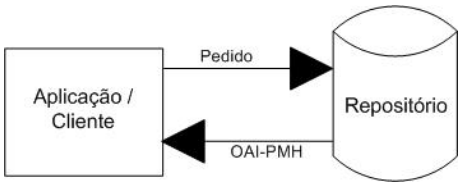

Figura 2.2: Pequeno exemplo do protocolo OAI-PMH

O protocolo tem duas características, a interoperabilidade e a extensibilidade.

A interoperabilidade é conseguida pela obrigatoriedade existente no protocolo na implementação do padrão de metadados, *Dublin Core*.

A extensibilidade é resultante da possibilidade de se criar ou utilizar outros padrões de metadados, já existentes ou não.

Quando os repositórios e os objectos digitais são criados desta maneira, o efeito global pode ser uma federação de repositórios que agregam conteúdos com atributos muito diferentes, mas que podem ser tratados da mesma forma devido às suas *interfaces* partilhadas [42].

Como foi dito anteriormente o uso mais comum deste *software* é maioritariamente para bibliotecas quer académicas quer de pesquisa.

Temos como exemplo a Biblioteca Nacional da Finlândia, Doria [2] ou a Universidade de Londres [3], TLRP, Reino Unido.

2.1.2 *EPrints*

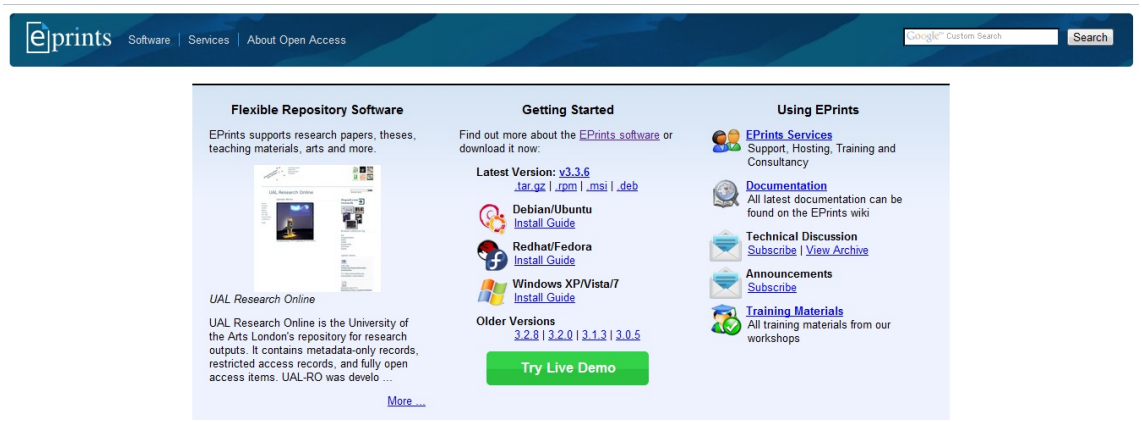

Figura 2.3: Página principal do *EPrints*

O *EPrints* foi desenvolvido pela ”School of Electronics and Computer Science”da Universidade de Southampton, Reino Unido [14] e é possível visualizar a sua página principal em 2.3.

Apesar de estar desenvolvido na linguagem *PERL*, é um *software* que é fácil de ser instalado. Assim como o *DSpace*, o *EPrints* também é um *software* ”out-of-the-box”, não é necessário possuir conhecimentos profundos de informática para poder ter o *EPrints* num computador.

O *software* foi inicial e expressamente concebido para a publicação de recursos científicos, levando a que o sistema fosse baseado na formação de páginas estáticas, *HTML*. Desta forma não era suposto que objectos digitais fossem inseridos ou actualizados muito frequentemente. Quando um objecto digital fosse inserido ou actualizado no repositório, todas as alterações eram visíveis só depois de uma reformulação da página estática. Como resultado final ocorre um atraso entre a edição e a publicação de um objecto digital.

No entanto, tem uma particularidade importante que outras bibliotecas não têm. É a sua capacidade em suportar vários ”arquivos”ou repositórios, sobre a mesma instância

de *EPrints*. Ou seja, este ”arquivo”pode ser visto como um conjunto de dados que pertencem ao mesmo assunto. Estes arquivos contêm os ficheiros, os *metadados* respectivos e podem ter a sua própria estrutura. Por exemplo, podemos ter um arquivo que contenha dados sobre matemática e outro arquivo com dados sobre estatística. O *EPrints* permite o armazenamento de vários tipos de ficheiros como por exemplo, imagens, dados de investigação, áudio, tudo o que seja conteúdo digital.

O *EPrints* torna-se importante quando se pretende definir os nossos próprios *metadados*. É um sistema flexível que não impõe um formato único para descrever os dados que estão contidos no repositório.

Para o *EPrints* um objecto digital é visto como um *item*, este tanto pode conter *metadados* como ficheiros.

O *EPrints*, assim como o *DSpace*, suporta a interoperabilidade entre as diferentes bibliotecas digitais.Este é o que melhor representa os padrões e ideais da *Open Archives Initiative* [37].

Apenas três tipos de acesso são suportados no sistema *EPrints*, o de autor , o de editor e o de administrador.

O de autor permite o *upload* de dados no sistema. No entanto, a funcionalidade terá que passar por um processo de autorização. Ou seja, o conteúdo só é guardado quando o editor aprovar o documento. O autor também pode actualizar a versão dos seus documentos e comentar o trabalho de outros autores.

O de editor permite a revisão do conteúdo de um documento, e a validação e a correcção do conteúdo dos metadados de um documento. O editor pode também corrigir ou excluir dados no repositório e visualizar ou alterar o registo de utilizadores no respositório.

O de administrador está encarregue, além da gestão do desempenho e acompanhamento do desenvolvimento da utilização do sistema, poder também adicionar autores e editores.

Uma das desvantagens deste sistema é, de facto, não permitir que sejam adicionados novos tipos de acesso. A explicação rebate na mesma questão, referida anteriormente, o facto de o sistema ter sido inicialmente desenvolvido com o intuito de publicar recursos científicos. Este problema culminou num sistema em que o *upload* se torna numa tarefa complicada e difícil de usar. O *EPrints* torna-se mais robusto quando o *upload* envolve pequenas quantidades de *items* com grandes volumes de informação.

Um exemplo da utilização do *EPrints* é o repositório de dados eCrystals – Southampton [11].

Figura 2.4: Página principal do *Fedora Commons*

2.1.3 *Fedora Commons*

O *Fedora Commons* [17] tem a assinatura de *Flexible Extensible Digital Object Repository Architecture* e não deve ser confundido com o sistema operativo *Fedora, Linux* [28].

O desenvolvimento do *Fedora Commons* foi efectuado pela Universidade de Cornell em parceria com a Universidade de Virgínia.

Ao contrário das outras duas bibliotecas digitais, o *Fedora Commons* é um *software* um pouco complexo e a sua instalação pode não ser muito trivial. Não é um produto chamado de ”pacote”, ou seja, pronto a ser instalado e usado. No entanto, este *software* dispõe de uma vasta e detalhada documentação para utilizadores menos experientes neste tipo de sistemas, o poderem instalar.

O *Fedora Commons* destaca-se das outras bibliotecas digitais, por ser muito flexível. Apresenta uma arquitectura onde a *interface* de utilizador, *front-end*, está separada do repositório. Este repositório permite o armazenamento e o acesso aos conteúdos digitais, como qualquer outro repositório descrito (*EPrints e DSpace*).

Apesar de parecer um *software* um pouco ”despido”, torna-se útil quando se pretende personalizar o *front-end* para fazer face às necessidades dos utilizadores.

Para a gestão de todo o conteúdo do repositório, o *Fedora Commons* apresenta uma *interface* de administração (figura 2.5), concebida na linguagem *Java*. Todo o sistema do *Fedora Commons* foi implementado na linguagem de programação *Java*.

Uma das vantagens que o *Fedora Commons* tem, é o facto de o sistema ser baseado nos princípios dos Serviços Orientados à Arquitectura (SOA). Ou seja, o *Fedora Commons* dispõe de serviços, que são expostos como *web services*.

Web Services é uma solução utilizada na integração de sistemas e na comunicação entre aplicações diferentes. Com esta tecnologia é possível que novas aplicações possam interagir com aquelas que já existem e que sistemas desenvolvidos em plataformas diferentes sejam compatíveis. Os *web services* são componentes que permitem às aplicações enviar e receber dados em formato XML. Cada aplicação pode ter a sua própria linguagem

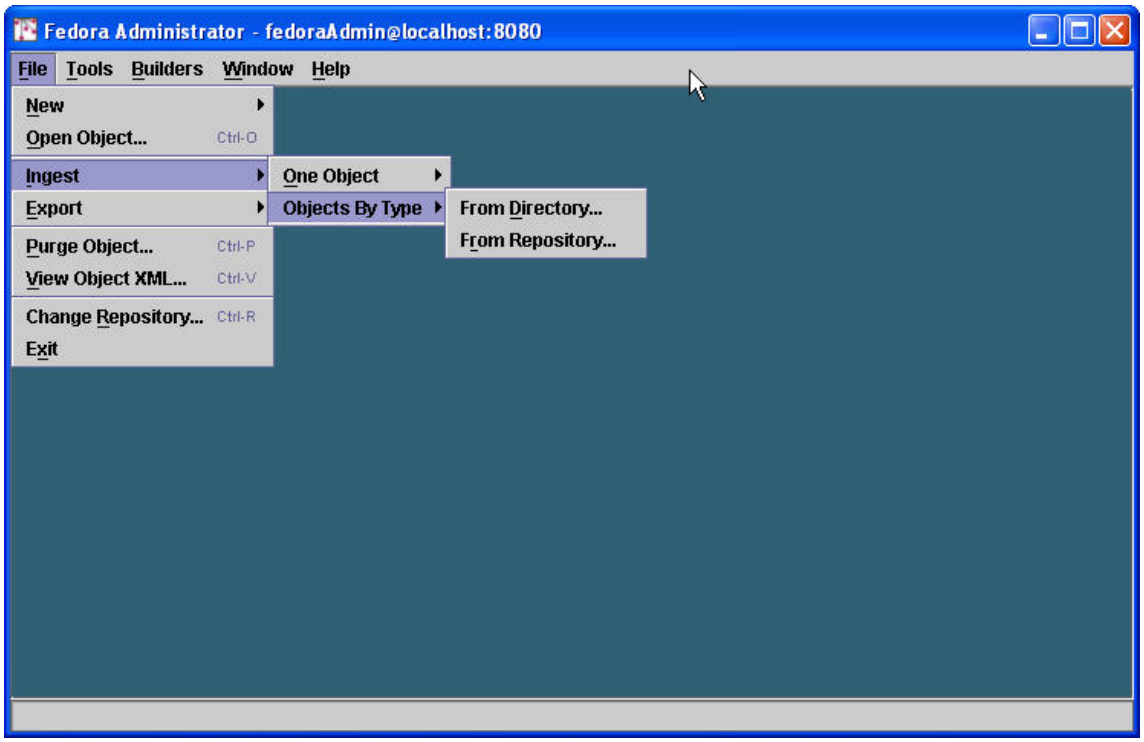

Figura 2.5: Aplicação administrativa, em *Java, do Fedora Commons*

que é traduzida para uma linguagem universal, o formato *XML*.

A utilização dos *web services*, por parte do *Fedora Commons*, permite não só a integração de outros sistemas, bem como a comunicação entre aplicações diferentes.

Todas as funcionalidades do *Fedora Commons*, a administração do repositório e o acesso aos objectos digitais são disponibilizados pelo ”repositório de serviços”, que é o núcleo central do *Fedora Commons*. Este providencia operações de registo e descoberta de todos os *web services*.

A implementação desta arquitectura permite a construção de bibliotecas digitais distribuídas, que são flexíves e permitem uma fácil personalização das suas funcionalidades com poucos custos de implementação.

Aliado aos serviços do repositório, o *Fedora Commons* também permite a definição de um modelo para o objecto digital. Na figura 2.6 é possível visualizar esse modelo.

O objecto digital, na perspectiva do *Fedora Commons*, é constituído por:

- **I**dentificador do Objecto Digital: um objecto digital é único e persistente, é-lhe atribuído um identificador, que é único no repositório, intitulado de PID (*Persistent unique IDentifier*). Para um dado objecto podem existir várias representações, ou seja, um livro pode ter o formato de um documento *.doc* ou *.pdf*.

- **C**omponente Descritiva: esta componente tem por objectivo descrever o objecto digital assim como a relação que existe entre este objecto e os outros que estão

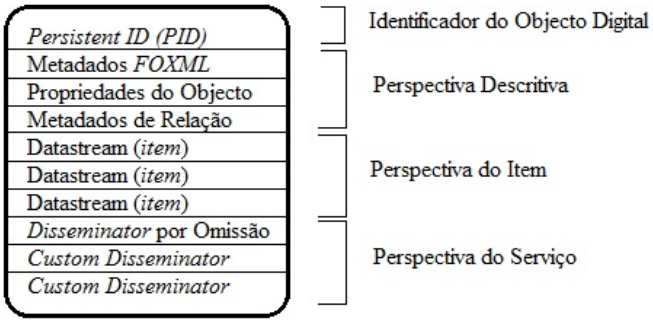

Figura 2.6: Objecto Digital visto pelo *Fedora Commons*

contidos no repositório. É constituída por três camadas, *metadados FOXML*, propriedades do objecto e medadados de relação.

O *FOXML (Fedora Object XML)* é um metadado diferente dos outros. Este é obrigatório para a arquitectura do repositório *Fedora Commons*. Os outros metadados são vistos como opcionais e destinados aos dados.

As propriedades do objecto referem-se ao tipo de objecto, ao estado, ao modelo do conteúdo, à data da sua criação e da sua última modificação.

Os metadados de relação pretendem representar as relações que existem entre os diferentes objectos digitais no repositório. Por exemplo, e pegando novamente no exemplo do livro, podemos dizer que este é constituido por capítulos.

- **C**omponente *Item*: representa o conteúdo do objecto propriamente dito como, texto, imagem, áudio, etc. Este conteúdo do objecto está associado ao termo *datastream*, que tanto pode conter um fluxo de dados como uma referência para um objecto externo.

- **C**omponente serviço: é destinada à recuperação imediata do objecto no repositório e adicionado quando criado o objecto digital. Voltando ao exemplo do livro, ter um serviço que devolva todos os seu capítulos.

O *Fedora Commons* define alguns *datastreams* que são especiais, tais como *policy*, *audit trail* e *relations* .

O *datastream policy* define as políticas de autorização para os objectos digitais, quer para proteger a integridade do objecto como para controlar o acesso ao conteúdo do objecto. Estas políticas são definidas por meio da linguagem *XACML* (*eXtensible Access Control Markup Language*) [16].

O *datastream audit trail* serve para manter o registo de todas as mudanças de um objecto durante o seu ciclo de vida.

O *datastream relations* é constituído pelo conteúdo de um ficheiro do tipo RDF/XML [26]. Este tem o objectivo de definir as relações que existem entre os objectos que estão

contidos no repositório. A particularidade deste ficheiro é a sua definição de relacionamento entre objectos. As definições do RDF são consideradas como ”triplas”, em que existe o conceito de sujeito, de predicado e de objecto.

No exemplo referido à pouco era possível ver a relação entre o livro e os seus capítulos. O *Fedora Commons* interpreta esta relação com *tags* próprias do RDF. A figura 2.7 ajuda a perceber esta representação da relação.

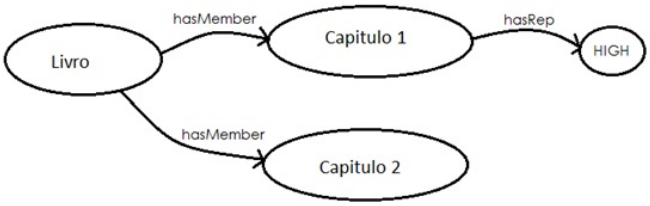

Figura 2.7: Exemplo de um gráfico RDF

Nesta figura temos um grafo direccionado onde os nós grandes são a representação dos objectos digitais e os nós pequenos são a representação desse objecto. Os arcos fazem a ligação entre os nós. Temos na figura os arcos *hasMember* e *hasRep*. O arco *hasMember* serve para dizer ao *Fedora Commons* que o livro é constituído por dois capítulos. Já o arco *hasRep* ajuda o *Fedora Commons* a perceber que tipo é o capítulo 1. Neste caso é uma imagem de tamanho grande (HIGH). Este grafo é transformado num ficheiro *RDF/XML*.

O *Fedora Commons* disponibiliza, entre outras, uma *interface* contida no repositório de serviços. Esta *interface*, intitulada de *Resource Index*, contém um grafo de relacionamentos que abrange todos os objectos digitais que estão contidos no repositório. Estes relacionamentos tanto podem ser implícitos pelo próprio modelo de objectos digitais como relacionamentos explícitos que são indicados no *datastream relations* do objecto. Este, além de guardar a informação dos relacionamentos, guarda também as propriedades do objecto digital e *datastreams* que contenham *DC*.

O *Resource Index* disponibiliza a funcionalidade de pesquisa sobre o grafo de relacionamentos através da linguagem iTQL [18] (*Interactive Query Language*). No código abaixo pode visualizar-se um exemplo de uma pesquisa sobre o *Resource Index*, neste caso pretende-se saber todos os objectos que tenham capítulo 1.

```
Select $object from <#ri>
where $object <rel:hasMember> <Capitulo1>
```

Para que todo o sistema que está por detrás do *Resource Index* funcione é preciso fazer a indexação de todos os objectos digitais que estão contidos no repositório. A indexação é realizada quando é feita uma inserção, modificação e eliminação de um objecto digital no repositório. Esta indexação está a cargo do *software Solr* [1] e pode ser visível na figura

2.8.

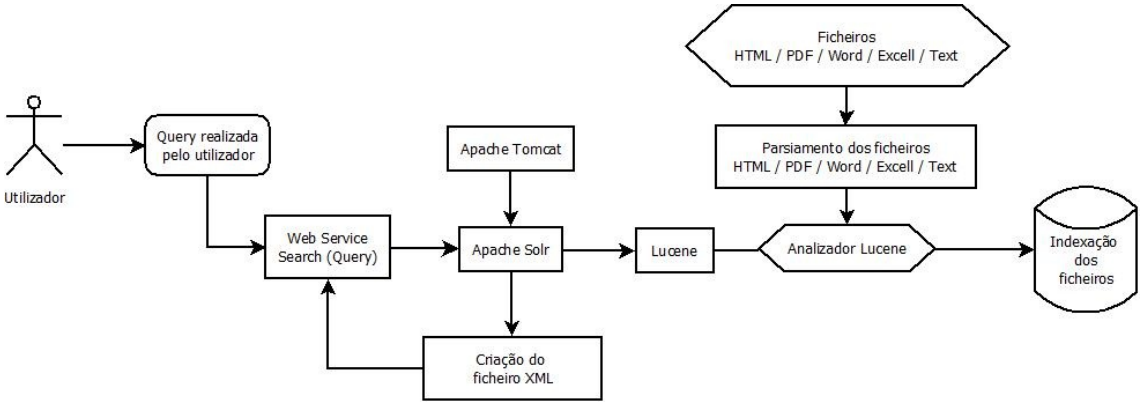

Figura 2.8: Pesquisa realizada no *Solr*

O *Solr* permite igualmente pesquisar sobre todo o conteúdo que existe no repositório, não só sobre ficheiros *RDF/XML*. Este contém uma linguagem própria para a realização da pesquisa e pode ser visualizado no código abaixo.

```
rel.hasMember: "Capitulo1" OR rel.hasMember: "Capitulo 1"
```

O *Fedora Commons* não difere das outras bibliotecas digitais e também ela oferece o suporte ao protocolo OAI-PMH.

Como exemplo da utilização do *Fedora Commons* temos a Universidade de Bibliotecas, *Rutgers* [27].

2.2 Sistema de Gestão de Conteúdo

> *"Nem sempre a Internet foi dinâmica, eficiente e útil como é actualmente" (Michellazo, 2007) [39]*

Há poucos anos atrás, o conteúdo da *Internet* era constituído por páginas estáticas, *HTML*. Procurava-se a todo o custo manter a página actualizada, quer ao nível de informação quer das novas tecnologias que iam surgindo. Era costume, nessa altura, haver pouco controlo sobre o *design*, navegação e autorização. A gestão da página estava a cargo de uma pessoa com conhecimentos avançados de informática. A pouco e pouco as pessoas iam deixando de fazer as alterações necessárias e a página ia ficando desactualizada ou com informação indisponível.

Actualmente as páginas são dinâmicas e personalizadas, trazendo os estilos e toques pessoais dos utilizadores para a Internet, remetendo a sua gestão para um sistema mais elaborado.

É neste contexto que surgem os Sistemas de Gestão de Conteúdo, *Content Management System*. Um SGC pode ser definido como uma base de dados de informação e uma forma de alterar e mostrar essa informação, sem despender muito tempo com detalhes técnicos da apresentação [43].

O *SGC* tem a particularidade de não restringir a linguagem de programação, possuir suporte para vários bancos de dados e sistemas operativos.

O principal objectivo de um *SGC* é estruturar, facilitar a publicação, disponibilizar, distribuir e administrar o conteúdo na Internet de forma fácil e intuitiva.

O *SGC* apresenta várias vantagens:

- Permite uma autonomia na actualização e edição de conteúdo sem que para isso sejam necessários conhecimentos avançados de informática;
- Redução do tempo na publicação, permitindo a disponibilização do conteúdo mais rapidamente *online*;
- É um produto não-pago e sem qualquer fidelização com alguma empresa que conceba páginas *web*;
- É incorporado num qualquer navegador da *web, Internet Explorer, Firefox, Chrome*, etc. Deste modo permite a realização de todo o processo de gestão deste, desde a sua criação até ao armazenamento de conteúdo;
- Maior flexibilidade para acrescentar ou editar módulos, ou seja, um *SGC* é constituído por módulos que podem ser personalizados à medida das necessidades;
- Maior segurança que permite a utilização de um sistema de autenticação de um utilizador;
- Permite o controlo de versões;
- Permite o uso da funcionalidade de comentar, deixando a cargo do utilizador a oportunidade de comentar um assunto numa página *web*.

Como em qualquer outra ferramenta existem requisitos que os CMS têm que cumprir:

- Separação do conteúdo da apresentação;
- Utilizadores, papéis (*roles*) e permissões, ou seja, todos os utilizadores que interagem com o sistema terão que estar autenticadas e autorizadas;
- Noção de contexto: o conteúdo é personalizado para o utilizador;

- Extensibilidade: o *SGC* deve providenciar uma *API* compreensiva e um sistema orientado a blocos que permita que os desenvolvedores possam alterar e extender o comportamento do sistema.

Alguns exemplos de SGC serão detalhados nas secções seguintes.

2.2.1 *Drupal*

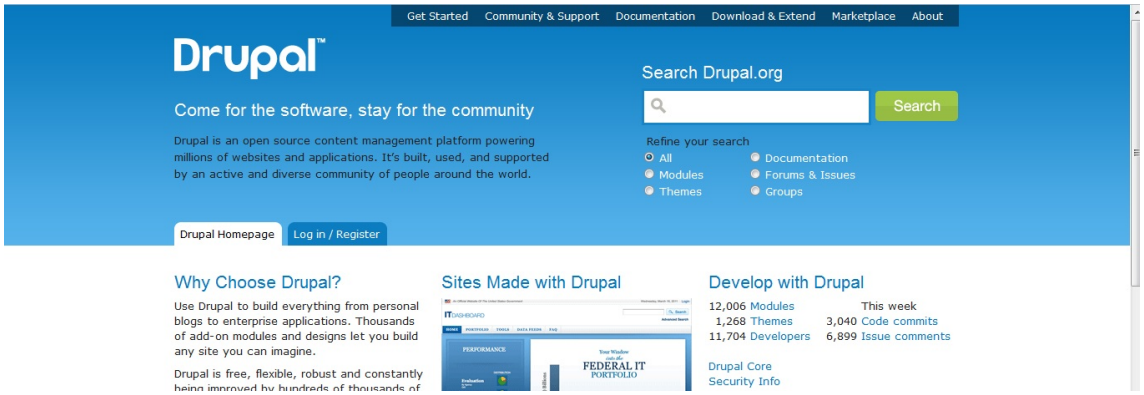

Figura 2.9: Página principal do *Drupal*

Em 2000, *Dries Buytaert* teve a ideia de criar um pequeno *site* de notícias, permitindo ao grupo de amigos deixar notas sobre o estado da rede, anunciar onde foram jantar ou outras notícias sem qualquer tipo de importância.

Só, em 2001, é que Dries decidiu lançar o *software* na *drop.org* como sendo o *Drupal* [6]. O objectivo inicial era permitir aos utilizadores o uso e aumentar a experiência para que outras pessoas pudessem explorar novos caminhos para o desenvolvimento.

Drupal é um *CMS* muito simples de se instalar e é muito flexível. Utiliza a base de dados *MySQL* ou *PostgreSQL* para armazenar o seu conteúdo e é escrito na linguagem de programação *PHP*. Pode ser visualizada a página principal em 2.9.

Dado que é um sistema muito flexível torna-se apetecível aos programadores, que desta forma podem personalizá-lo ao seu gosto. Em contrapartida os *designers* terão bastantes dificuldades em alterar o conteúdo no *Drupal*.

O *Drupal* funciona à base de módulos e estes são, sem dúvida, um conceito central de extensibilidade. Por exemplo, um módulo só para a parte da gestão de utilizadores e outro para a gestão de notícias na página *web*.

O facto de o *Drupal* ser constituído por módulos faz com que todo o sistema seja adaptável às evoluções da *Internet*.

Ao longo da existência do *Drupal* os utilizadores têm vindo a descobrir as suas vantagens e desvantagens, sendo que grande parte das vantagens recai em :

- Ser extremamente virado para os programadores;
- Ter uma boa comunidade que ajuda os programadores caso ocorra algum problema;
- Ser muito poderoso na criação de páginas *web ricas* em conteúdo dinâmico.

A principal desvantagem será o facto do sistema não ser *user-friendly*, ou seja os utilizadores com pouca experiência terão grande dificuldade em desenvolver as suas páginas *web* em *Drupal*.

Uma funcionalidade bastante importante do *Drupal* é a forma como lida com o controlo de acesso. Este utiliza dois mecanismos:

- Permissões, as quais são orientadas à acção do utilizador. Por exemplo, o utilizador pode editar, eliminar, ler, etc.
- Papéis (*roles*), são um conjunto de permissões que são atribuídas ao utilizador. Ou seja, o utilizador pode editar e eliminar um recurso.

2.2.2 *WordPress*

O *WordPress* [30] foi criado, em 2003, por Bryan Bonem e MathewMullenweg. É muito robusto, fácil de instalar e de usar, a figura 2.10 apresenta a página principal do *WordPress*.

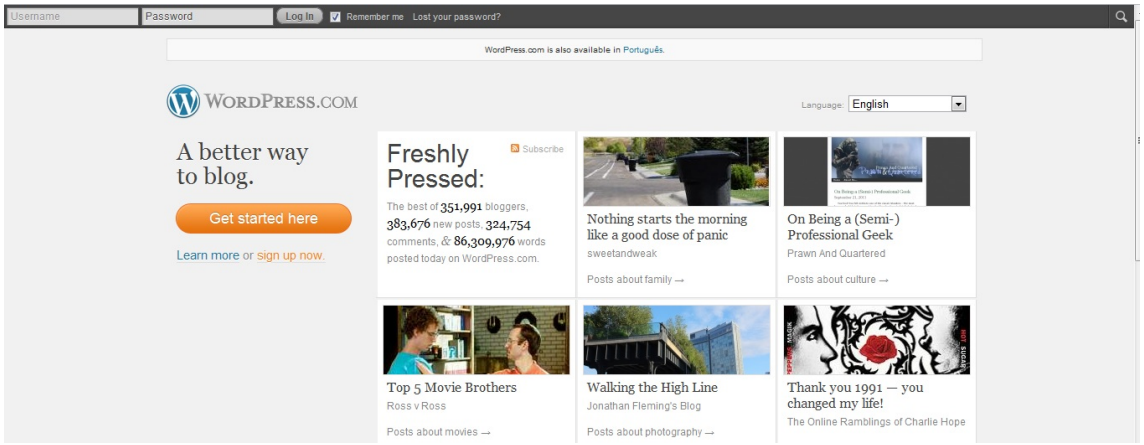

Figura 2.10: Página principal do *WordPress*

O *WordPress* é um *CMS* excelente para se usar na criação de páginas *web*. Ficou, no entanto, muito conhecido por ser um sistema de blogue, apesar de oferecer ferramentas para implementar outro tipo de sistemas como sites, pequenas lojas virtuais e *sites* de compras colectivas.

Como é usual ver-se num blogue, o*WordPress* oferece um sistema de comentários, onde os utilizadores podem deixar a sua opinião sobre o artigo publicitado. Esta funcionalidade favorece a interacção e o conceito de comunidade, ao invés de uma simples e estática página *web*.

Este *software* está fortemente equipado para ser um sistema de blogue, tornando desnecessário qualquer adição de funcionalidades ao sistema. No entanto, o *WordPress* tem a particularidade de permitir a adição de *plugins*, que são pequenos *scripts* que fornecem funcionalidades adiccionais ao blogue.

O *WordPress* permite também ao utilizador reorganizar o *layout* do seu blogue através de *widgets*, *interface* gráfica, sem que para isso este precise de editar o código *PHP* ou *HTML*.

O *WordPress* foi desenvolvido em *PHP* e é suportado por uma base de dados *MySQL*.

Uma das grandes vantagens do *WordPress* é a facilidade do seu uso dado que o utilizador torna-se cada vez mais capaz para usar o sistema.

A grande desvantagem deste sistema é a rapidez no ciclo de *upgrades*. É necessário manter sempre o *site* actualizado, independentemente de o utilizador o desejar ou não, por motivos de segurança.

2.3 Autorização e Autenticação em Sistemas de Informação

"To be or not to be, that is the question"(William Shakespeare)

Com o forte crescimento da divulgação de informação a sua distribuição torna-se fundamental para a privacidade dos utilizadores, a confidencialidade dos dados expostos e a integridade da informação.

A segurança da informação é essencial e deve ser mantida com métodos e políticas de controlo de acesso fortes e muito bem definidas e elaboradas. Desta forma os membros utilizadores podem depositar confiança nas interacções com os outros utilizadores e posteriormente vir a contribuir ainda mais para o sistema.

Normalmente são as organizações que definem as políticas de acesso no intuito de proteger os seus dados contra ameaças que possam comprometer a sua informação.

A segurança está fundamentada em quatro propriedades que devem ser garantidas [35]:

- Confidencialidade - a informação só pode ser revelada caso o utilizador esteja autorizado a acedê-la;

- Integridade - a informação não pode ser modificada por utilizadores que não possuam direitos para tal;
- Disponibilidade - o sistema não deve ser negado a utilizadores autorizados;
- Autenticidade - o sistema só pode ser acedido por utilizadores autenticados.

Sandhu e Samarati definem o controlo de acesso como o acto de limitar as operações que podem ser realizadas por uma entidade sobre um determinado recurso. Este permite alcançar os objectivos da segurança da informação prevenindo a exposição e modificação não autorizada da informação sensível [41].

Um sistema de controlo de acesso monitoriza os pedidos de acesso e implementa regulações (políticas), que definem quem pode ou não executar acções e em quais recursos. O sistema de controlo de acesso permite a especificação de políticas através de atributos/propriedades genéricas.

A linguagem que actualmente é a mais utilizada para definir políticas é a linguagem *XACML*. Esta permite especificar políticas de segurança, requisições e respostas para decisões de controlo de acesso. Desta forma o uso destas políticas faz com que as organizações possam controlar o acesso aos seus conteúdos e às informações protegidas.

A linguagem de políticas de controlo de acesso define quem possui os direitos e sobre o quê. O formato de pedido e resposta descreve como as consultas sobre o sistema de políticas deverão ser realizadas e como deverão ser as respostas. Este formato define as trocas entre o *Policy Decision Point (PDP)*, que processa a política, e o *Policy Enforcement Point (PEP)*, que realiza as decisões de políticas.

Pode-se verificar na figura 2.11 como se desenrola o pedido a um recurso.

A principal vantagem na utilização de políticas de controlo de acesso é o aumento da escalabilidade e flexibilidade. A escalabilidade prende-se com o facto da mesma política poder ser aplicada a um grande número de objectos. A flexibilidade pode ser ampliada pela separação entre políticas de controlo de acesso e a implementação do sistema de controlo de acesso, tornando-se numa vantagem nestes sistemas. Ou seja, é possível adaptar as políticas aos diferentes utilizadores bastando apenas editar essa politica. No entanto, não foi alterado a sua implementação.

Contudo, há que ter em mente um aspecto fundamental e muito importante.

Com o enorme sucesso dos sistemas de distribuição de serviços e disseminação da informação na *Internet*, a protecção da privacidade dos utilizadores é um requisito fundamental.

O conteúdo dos dados ou metadados existentes no repositório contêm informação sensível fazendo com que o acesso a esses dados seja limitado a utilizadores credíveis e pertencentes ao grupo desse repositório. Portanto, ter a privacidade dos utilizadores que utilizam um dado repositório é um requisito fundamental.

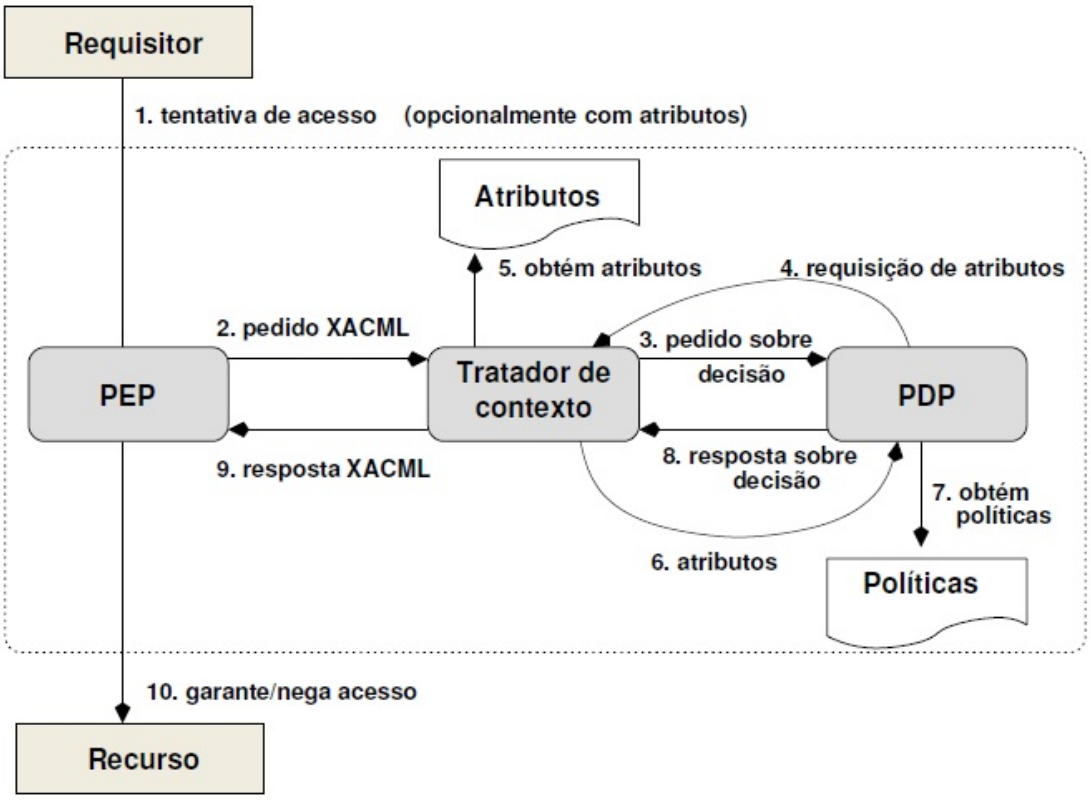

Figura 2.11: Processo de controlo de acesso

As características da segurança, mais uma vez, prevalecem quando ocorre a necessidade de protecção.

Como referido por Russell e Gangeni a autenticidade torna-se ponto-chave para colmatar esta lacuna [35].

Neste caso o repositório precisa de perceber quais são os utilizadores permitidos no sistema.

É neste contexto que surge o *OpenLDAP* (*Open Lightweight Directory Access Protocol*) [23], a sua página principal pode ser visualizada na figura 2.12. É um serviço de directórios baseado no protocolo LDAP onde contém toda a informação relativa ao utilizador, como *username, password*, nome, *e-mail*, etc.

Entenda-se directório como sendo uma base de dados que armazena dados de forma hierárquica e que contém mecanismos poderosos de pesquisa optimizados.

Como exemplo de um directório pode-se considerar o *DNS* (*Domain Name System*) [5]. Este contém a relação entre os nomes dos *hosts* e o seu respectivo endereço de *IP (Internet Protocol)*. Quando se pretende aceder a um *host* pelo nome e não pelo seu endereço IP existe um procedimento por trás, chamado de resolução de nomes, onde é feita uma pesquisa pelo nome do *host* ou endereço electrónico. Consideremos o *host* tão

Figura 2.12: Página principal do *OpenLDAP*

conhecido como o *Google*. O endereço pode ser resolvido por meio de uma pesquisa na ”árvore”parecida à da figura 2.13. É realizado primeiro uma pesquisa sobre o último pedaço do endereço do nome do *host*, ou seja, .com de seguida por .google e por fim .www.

O serviço de directórios é então projectado para a gestão de entradas e atributos num directório. Este permite a centralização da informação relativa ao utilizador e oferece:

- Flexibilidade - permite a diversidade de informação como a informação sobre o utilizador e sobre a organização a que este pertence, por exemplo;
- Segurança - possui mecanismos de autenticação;
- Escalabilidade e adaptabilidade - adapta-se à estrutura da rede actual e possíveis alterações futuras;
- Extensibilidade - permite alterações na rede.

O *OpenLDAP* além de implementar as características que os serviços de directório oferecem, reforça também o conceito de segurança. Ou seja, a comunicação que existe entre a aplicação/cliente e o directório, que contém a informação dos utilizadores, tem suporte ao TLS (*Transport Layer Security*) que faz a criptografia dessa comunicação. Além disso, o *OpenLDAP* utiliza vários algoritmos de criptografia como *SHA-1* ou *MD5* para armazenar as *passwords* dos utilizadores.

O *OpenLDAP* é muitas vezes utilizado por sistemas que queiram implementar um serviço de *Single Sign On* (SSO). Ou seja, um sistema que contém uma autenticação única facilitando assim a interacção com os outros serviços como o serviço de *e-mail*.

Embora seja possível ter o directório remotamente (escalabilidade), o *OpenLDAP* não é um serviço muito usado na *Internet*, apenas em *Intranet* (redes particulares), principalmente em grandes empresas. Este torna-se útil quando existe um maior número de utilizadores e de documentos partilhados entre eles.

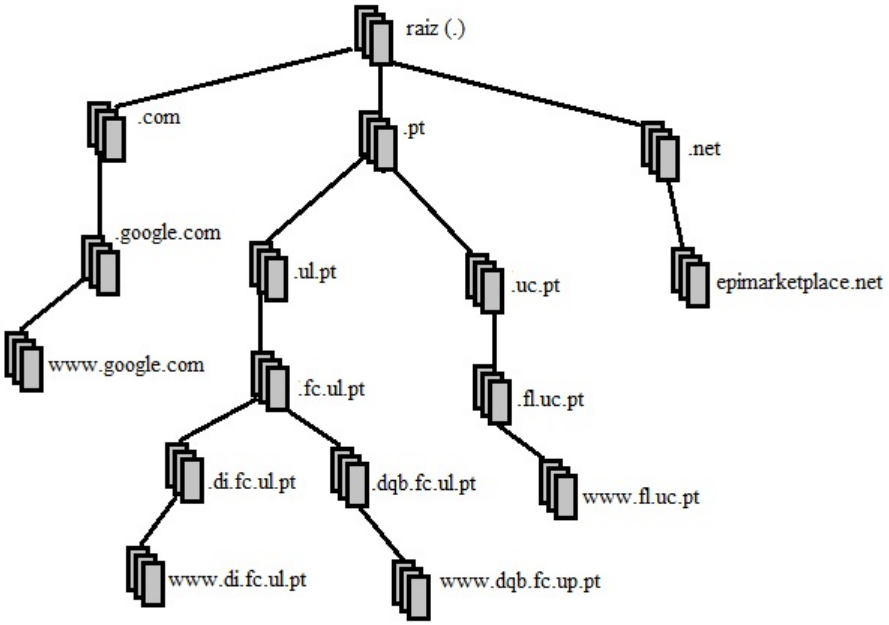

Figura 2.13: DNS, exemplo de um Directório

O modelo de informação do *OpenLDAP* é baseado em atributos onde o nome distinto (*Distinguished Name ou DN*) é único. O uso do DN permite a identificação e diferenciação de uma entrada no sistema de directórios. Cada atributo tem um tipo e um ou mais valores. Por exemplo o atributo telefone pode conter o número de um telemóvel como também o número de um telefone fixo. Na figura 2.14 pode visualizar um exemplo de uma entrada no *OpenLDAP*.

Uma das grandes vantagens, além da optimização da pesquisa, é a replicação dos dados contidos no directório. Apesar do conteúdo estar centralizado é possível replicar a informação por vários servidores evitando assim casos de catástrofe ao eliminar-se acidentalmente um directório.

A principal desvantagem neste sistema é possívelmente a sua instalação onde contém pré-requisitos como o *OpenSSL, Kerberos, SASL(Cyrus) e Berkeley DB*.

Uma outra particularidade do *Fedora Commons* é o facto de disponibilizar ao utilizador a ferramenta *FeSL - Fedora Security Layer*. Este reúne no mesmo *software* a oportunidade de se realizar a autenticação e autorização. Oferece uma camada de autenticação permitindo uma redução na complexidade e permite ainda a integração com outros sistemas de autenticação. Esta camada foi implementada em *Java Authentication and Authorization Service (JAAS)*. O *FeSL* também contém uma camada de autorização com base no *Fedora - XACML*, ou seja, implementa o mesmo raciocínio que as políticas *XACML*, neste caso, permite o controlo de acesso possa ser ao nivel de uma colecção, objecto digital ou até mesmo ao nível do *datastream*. Este *software* torna-se útil porque reúne o que melhor podemos querer em questões de autenticação e autorização, ou seja, podemos

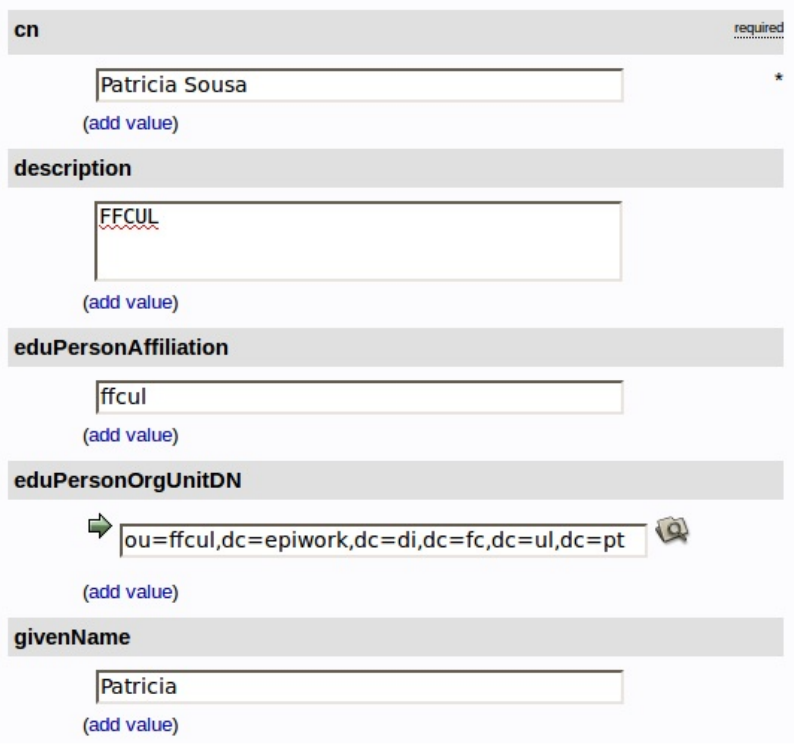

Figura 2.14: Um exemplo de uma entrada no *OpenLDAP*

ter a integração de políticas *XACML* com autenticação externa. Tudo isto torna qualquer sistema muito mais leve, o preferido de qualquer programador com necessidade de ter um sistema com autenticação e autorização.

2.4 Serviços *Web*

"The web is simply defined as the universe of global network-acessible information."(Tim Berners-Lee)

Segundo Berners-Lee, o primeiro objectivo da *web* é estabelecer um espaço em que a informação seja partilhada. Em que os sistemas possam participar com a publicação de recursos e serviços nesse mesmo espaço. Desta forma reflecte-se na visão do *W3C* de um futuro para a arquitectura da *web* [33].

A necessidade da integração entre aplicações e a unificação de processos em diferentes sistemas e escritos em diferentes linguagens, fez com que se criasse os serviços *web*. Permitindo assim, disponibilizar formas de integrar sistemas distintos, modularizar serviços e capacitar a integração e consumo de informações.

Segundo a definição do *W3C*, um serviço *web* é um sistema designado para suportar interoperabilidade entre máquinas sobre a rede. Ou seja, permite que programas escritos em diferentes linguagens e em diferentes plataformas se comuniquem através da rede [47].

A *World-Wide Web*, ou simplesmente *Web* é um espaço de informação em que os objectos de interesse, chamados de recursos, são identificados por chaves globais chamadas de *Uniform Resource Identifiers* (URI).

Segundo Berners-Lee, o sistema de identificação URI disponibiliza uma forma simples e extensível de identificar recursos, como páginas *Web*, imagens, vídeos e serviços. O formato HTML foi o primeiro formato de dados usado na Web. Desde então, vários formatos surgiram para a representação de recursos. Segundo Fielding o protocolo HTTP (*Hypertext Transfer Protocol*), é um protocolo voltado a sistemas distribuídos, colaborativos e hipermídia [36].

Um serviço *web* pode ser considerado como a implementação de um Agente concreto, em que este é um *software* ou *hardware* que envia e recebe mensagens, enquanto que o serviço é o recurso caracterizado por um conjunto abstracto de funcionalidades que é fornecido pelo Agente.

O objectivo dos serviços *web* é providenciar várias funcionalidades aos diferentes utilizadores. Estes utilizam o protocolo *HTTP* e um documento, *XML* ou *JSON* como resposta ao serviço. Ou seja, cada aplicação pode ter a sua própria ”linguagem”, que é traduzida para uma linguagem universal, como é o caso do formato *XML*.

O *W3C* também refere que se pode identificar duas grandes classes de serviços *web*:

1. *REST* - o objectivo é manipular representações de *XML* dos recursos, utilizando um conjunto de operações de ”estado”;

2. *SOAP* - conjunto arbitrário de serviços *web* em que o serviço pode expor um conjunto de operações.

REST

REST é o acrónimo para *Representational State Transfer* e a sua arquitectura está relacionada com a Arquitectura Orientada ao Serviço (SOA - *Service-Oriented Architecture*).

A implementação de um serviço como o *REST* segue quatro principios básicos de *design* [44]:

1. utilização de métodos *HTTP*: *GET, POST, PUT* e *DELETE*;

2. não tem estado;

3. expor uma estrutura como ”caminho de directorias”nos *URIs*;

4. transferir *XML* ou *JSON* ou ambos.

O conceito mais importante da Arquitectura Orientada ao Serviço é o ”recurso”. Um recurso é identificado pelo *URI*. Este é uma *string* que pode ter grande significado e muito bem entendido pelos utilizadores. É ao fim ao cabo um vocabulário que pode ser partilhado entre o serviço e os seus consumidores, ou seja, utilizadores.

O utilizador apenas especifica a representação do que pretende na *string URI*, ou seja, estes *URIs* determinam como um serviço *REST* é intuítivo e como o serviço vai ser usado. Portanto, estes *URIS* devem ser construídos de forma a serem fáceis de se advinhar o que se pretende dos serviços *web*. A *string URI* inclui um conjunto de paramêtros que definem um critério para utilizar os diferentes serviços disponíveis aos utilizadores.

Já se percebeu que a *URL* é a representação de um objecto que pode ser obtido usando o método *GET* do protocolo *HTTP*, pode ser removido por utilização de *DELETE* ou pode ser modificado com o método *POST* ou *PUT*.

Podemos pensar num exemplo com o sistema *Twitter*. Suponhamos que se pretende pesquisar no *Twitter* por *H1N1*. É interessante constatar que o *Twitter* oferece um serviço de pesquisa muito bem definido e para isso apenas se utiliza a seguinte *URI*:

```
https://twitter.com/search/H1N1
```

Ao inserir este *URI* no *browser*, o que se está a fazer é dizer ao *twitter* que queremos obter o resultado da pesquisa por *H1N1* e para isso utilizamos o método *GET* do protocolo *HTTP*. O resultado desta pesquisa é um *XML* que depois é tratado pelo *front-end*, *twitter* e mostrado ao utilizador por forma a ser mais compreensível.

A idéia central neste sistema é a utilização de um URI de identificação única para cada recurso. Dependendo do método utilizado para invocá-lo e dos dados na requisição HTTP, este URI terá um funcionamento diferente.

Segundo Fielding existe um conjunto de restrições que devem ser seguidas quando se pretende implementar serviços *web* em *REST*, são eles [44]

1. cliente-servidor: ocorre a separação entre a estrutura do utilizador e a estrutura do servidor. O servidor, com um conjunto de serviços espera por requisições aos seus serviços. O utilizador que deseja que um serviço seja executado, envia uma requisição ao servidor. Este tanto pode rejeitar como executar o serviço solicitado e retornar a resposta ao utilizador. Esta separação permite que os dois componentes, utilizador e servidor, possam evoluir independentemente suportando o requisito de escalabilidade da *Internet*;

2. sem estado: A comunicação entre utilizador e servidor deve ser feita sem o armazenamento de qualquer tipo de estado, ou seja, cada requisição do utilizador para o

servidor deve conter todas as informações necessárias para que ela seja entendida. Portanto, estados de sessão, quando necessários, devem ser totalmente mantidos no cliente;

3. *cache*: O facto de ser sem estado, traz desvantagens, a redução de desempenho na rede é uma delas. Uma forma de diminuir este impacto é a utilização de *cache*. Com a utilização de *cache* torna-se possível diminuir ou até mesmo eliminar completamente algumas interações com o servidor, optimizando a eficiência, a escalabilidade e o desempenho para o utilizador;

4. *interface* uniforme: A característica central que diferencia o estilo arquitetural REST de outros estilos baseados em rede é sua ênfase em uma *interface* uniforme entre os componentes utilizador e servidor. Com o objectivo de obter uma *interface* uniforme, REST define quatro requisitos de interface: Identificação de recursos, manipulação de recursos através de representações, mensagens auto-descritivas e hipermídia como mecanismo de estado da aplicação;

5. sistema em camadas: Sistemas em camadas utilizam camadas para separar diferentes unidades de funcionalidade. A principal desvantagem deste modelo está no aumento de *overhead* e latência nos dados processados, reduzindo o desempenho.

6. *code-on-demand*: este é uma restrição opcional. REST permite que utilizadores tenham a funcionalidade de fazer *download* e executar diretamente o código no lado do utilizador. Um exemplo prático conhecido é o *Adobe Flash*: Um utilizador solicita uma página *Web* a qual contém um *link* para um *SWF* usando um *browser*. Após a requisição, a página *Web* é transportada para a máquina do cliente juntamente com um *SWF* e o mesmo é executado.

SOAP

SOAP, *Simple Object Access Protocol* , é a especificação de um protocolo para a troca de dados entre dois *endpoints* [40]. Ao contrário do *REST*, a especificação deste serviço nuca adoptou a noção de partilha de informação no espaço com o uso de *URI*. Estes presumiram que cada aplicativo devia definir o seu próprio e único espaço apartir do zero com uso ao *WSDL*. Este descreve apenas um só recurso *web* e não providência a descrição de *links* para outros recursos. Assim o *SOAP* e o *WSDL* utilizam as *URIs* expressamente para *endpoints*.

O *SOAP* consiste em três partes:

1. A construçao do envelope *SOAP* define uma *framework* para expressar o que está na mensagem, quem deve receber o envelope e se é mandatório ou opcional;

2. A definição de regras de *encoding* permite um mecanismo de serialização que pode ser usada para troca de instâncias de tipos de dados definidos nas aplicações;

3. A representação de *RPC SOAP* define a conversão que pode ser usada para representar *remote procedure calls* e respostas.

Um ponto interessante a referir perante estes dois serviços *web* será o comportamento em termos de segurança. Com o serviço *web REST* qualquer *firewall* consegue perceber a intensão de cada mensagem através da análise dos comandos *HTTP* usados no pedido. Em contra partida o pedido do *SOAP* utiliza o comando *POST* como forma de comunicação e um encapsolamento de mensagens, desta forma é impossível à *firewall* perceber se o conteúdo do pedido é apenas para realizar um pedido ou, por exemplo, eliminar as tabelas de uma base de dados. Neste sentido é mais seguro o serviço *web REST* com o acrescento de se poder utilizar o protocolo *SSL* dando mais segurança ao serviço. O *SOAP* para combater esta lacuna terá que utilizar a encriptação nas suas mensagens levando a que utilizadores sem conhecimentos informáticos tenham mais dificuldade em implementar esse sistema.

Como nota final, pode-se concluir que o *SOAP* requer uma implementação cuidaa e um esforço maior por parte do utilizador. Enquanto que o *REST* requer grande esforço de implementação por parte do servidor.

Quando se pensa em implementar serviços *web* deve-se ter em conta a facilidade com que os desenvolvedores podem aceder ao sistema com uma forte documentação.

É a complexidade no lado do utilizador que conta, levando a que eles utilizem ou não o nosso sistema.

2.5 *Software* Utilizado no *Epidemic Marketplace*

Do *software* aqui descrito procedeu-se a uma selecção, com base não só nos requisitos da plataforma como na curva de aprendizagem que estes *sofwtares* exigiam.

A escolha final, perante as diferentes hipóteses, foi a seguinte:

1. Bibliotecas Digitais: *Fedora Commons*. A escolha recaiu pelo simples facto de ter sido, foi utilizado na versão 1.0. A sua flexibilidade permitiu com que fosse possível criar o modelo do *EM* de forma simples. Todos os outros, *softwares* da mesma categori, apresentavam dificuldades em concretizar este requisito;

2. Sistema de Gestão de Conteúdos: *Drupal*. Foi o que melhor se destacou, não só porque permitiu uma maior facilidade em conter um sistema de papéis personaliza-

dos como também por ser desenvolvido na linguagem *PHP*. Desta forma, a curva de aprendizagem era muito pequena visto eu já ter conhecimentos nesta linguagem;

3. Autenticação e Autorização: *XACML* e *OpenLDAP*. Como referido no ponto anterior, a versão 1.0 já continha estes dois *softwares* por isso a curva de aprendizagem era quase nula;

4. Serviços *Web*: *REST* em linguagem *Python*. Este foi sem dúvida o maior desafio, uma vez que não tinha conhecimentos para realizar serviços orientados para a web na ferramenta *python*.

Capítulo 3

Requisitos do Epidemic Marketplace

O *Epidemic Marketplace* (EM) pode ser definido como um repositório virtual distribuído, uma plataforma com suporte à transparência, acesso distribuído, recursos heterogéneos e redundantes (Kuliberda *et al.* 2006, Ohno-Machado *et al.* 1997).

É um repositório virtual, porque os dados podem ser armazenados em sistemas que são externos ao EM, e fornece acesso transparente pois várias heterogeneidades estão escondidas dos seus utilizadores.

Este capítulo descreve os requisitos gerais e outros diversos que foram propostos para o EM. Todos estes requisitos foram retirados do *D3.3 Public Release of the Epidemic Marketplace Platform* disponível em [48]

1. Requisitos de Sistema
2. Requisitos de *hardware* para a implementação do Sistema
3. Requisitos Não Funcionais
4. Requisitos Funcionais

3.1 Requisitos de Sistema

Vários requisitos foram identificados ao definir a arquitetura do EM. Esses estão directamente relacionados com os objectivos do Projecto *Epiwork* e estão abaixo descritos.

3.1.1 Suporte e partilha e gestão de conjunto de dados epidemiológicos

Os utilizadores registados devem ser capazes de carregar e avaliar conjuntos de dados anotados. O conjunto de dados anotados, irá compor um catálogo que estará disponível para os utilizadores.

3.1.2 Suporte de integração perfeita de fontes de dados heterógeneos

Os utilizadores devem ser capazes de ter uma visão unificada das fontes de dados relacionados. Os dados devem estar disponíveis a partir de fontes de *streaming*, estáticas e dinâmicas. Todos os dados recuperados pelos utilizadores ou outros serviços devem estar disponíveis através de uma *interface* comum.

3.1.3 Suporte à criação de uma comunidade virtual para a investigação epidemiológica

A plataforma servirá como um fórum de discussão que vai orientar através da descoberta das necessidades de partilha de dados entre os fornecedores e os modeladores. Os utilizadores tornar-se-ão participantes activos, gerando informação e fornecendo dados para a partilha e colaboração *online*.

3.1.4 Arquitetura Distribuída

O EM deve implementar uma arquitetura distribuída geograficamente em vários *sites*. A arquitectura distribuída deve fornecer uma melhoria de acesso aos dados, uma maior disponibilidade e tolerância a falhas.

3.1.5 Suporte ao acesso seguro aos dados

O acesso aos dados deve ser controlado. O EM deve fornecer o *single sign on*, e as políticas de autorização de acesso múltiplo, personalizado pelos utilizadores. Todos os *sites* do EM devem depender de uma infraestrutura distribuída de autenticação.

3.1.6 Suporte de análise de dados e simulação em ambientes *grid*

O EM irá fornecer serviços de análise de dados e de simulação em ambientes *grid*. Portanto, o EM deve operar perfeitamente com serviços específicos de *grid*, tais como, serviços de segurança e de alocação de recursos.

3.1.7 Fluxo de Trabalho

A plataforma deve apoiar fluxos de trabalho para processamento de dados e serviços externos de interacção. Este requisito é particularmente importante para aqueles serviços que recuperam, processam e armazenam os dados processados no EM, tais como as grelhas de análise e serviços de simulação.

3.2 Requisitos de *Hardware*

Os diversos módulos do EM estarão disponíveis *on-line* e devem estar preparados para serem usados por uma vasta comunidade de utilizadores e manipular grandes conjuntos de dados. Um requisito importante é que este sistema seja estável e esteja disponível todo o tempo, por isso deve ter uma poderosa e estável base de *hardware*.

Para a instalação local analisámos os requisitos e decidimo-nos pelo uso de dois servidores, complementados com duas unidades de armazenamento em rede (*Netwok storage units*). Este sistema proporciona um bom nível de redundância e faz a recuperação de falhas possíveis de forma rápida e fácil. Essa configuração permite o uso de um servidor para fins de teste, mantendo o outro com uma versão estável em execução.

A conectividade deve ser fornecida inicialmente por um *link multi-gigabit* compartilhada entre a Universidade de Lisboa e o GÉANT, a rede europeia de investigação e ensino superior. O acesso ao GÉANT vai garantir a conectividade entre o *site* principal do EM e outros parceiros Epiwork.

3.3 Os requisitos Não Funcionais

Os principais requisitos não funcionais que foram identificados para o EM são os seguintes:

3.3.1 Interoperabilidade

O EM deve interagir com módulos que são desenvolvidos pelos outros WPS, como a plataforma de monotorização Influenza do WPS e a plataforma de simulação do WP4. No futuro, sistemas desenvolvidos por investigadores através do mundo podem precisar de questionar o catálogo do EM ou ter acesso ao conjunto de dados disponível no EM e é deste modo que deve ser suportada a comunicação entre as diferentes tecnologias.

3.3.2 Modularidade

Nem todos os locais onde o EM será implantado precisam de ter todos os módulos instalados. Por exemplo, alguns *sites* podem não precisar do módulo colector de dados, portanto a modularidade é um requisito importante do EM.

3.3.3 *Open-source*

Todos os pacotes de *software* usados na implementação e instalação do EM devem ser de *open source*, bem como os novos módulos desenvolvidos especificamente para o EM. Uma solução *open source* reduz o custo de desenvolvimento, melhora a confiabilidade do *software* e simplifica o suporte.

3.3.4 Baseado em Padrões

A fim de garantir a interoperabilidade entre o EM e o *software* desenvolvido por outros WP's, bem como uma perfeita integração de todos os *sites* geograficamente dispersos do EM, o sistema deve ser construído sobre serviços *web* (*web services*), autenticação e padrões de metadados.

3.4 Requisitos do Repositório

O repositório guarda e preserva colecções de dados, para serem usados e partilhados activamente pelos utilizadores da plataforma e automaticamente recuperados pelo módulo colector.

3.4.1 Separação de dados de metadados

Uma característica importante da arquitectura para repositórios científicos em geral, e também do EM, é a clara separação entre dados e metadados (Stolte *et al.* 2003).

Por exemplo, deve haver uma separação clara entre os metadados e esquemas de dados actuais, dado que os metadados podem conter informações não disponíveis directamente em esquemas de dados.

3.4.2 Suporte para os padrões de metadados

É necessário um amplo suporte para padrões de metadados para recursos *web* de gestão e processamento (procura, por exemplo). Isto significa a adopção de *Dublin Core* (*Dublin Core* 2009).

Devido a restrições de privacidade é possível que apenas os metadados de algumas fontes de dados estejam disponíveis através do EM,. Nestes casos, o utilizador deve obter os dados directamente do *site* que hospeda a fonte de dados, directrizes a seguir descritas no *Epidemic Marketplace*.

3.4.3 Suporte à Ontologia

Um passo adiante na implantação do EM foi ter um repositório semanticamente habilitado usando ontologias para descrever e estruturar os dados (Goni *et al.* 1997, Fox *et al.* 2006). O EM providenciará uma *framework* para a criação e desenvolvimento de ontologias, para responder abertamente às necessidades desta comunidade e fomentar a sua participação activa. Nós começamos por usar ontologias existentes, tais como o Sistema Único de Linguagem Médica (UMLS) (Bodenreider 2004). A ontologia geográfica, com cobertura mundial, é também uma necessidade primária (Chaves *et al.* 2005). O nosso objectivo é contribuir para tornar as ontologias amplamente aceites pela comunidade Epidemiológica

e garantir a sua evolução sustentável, através da replicação bem sucedida de iniciativas semelhantes, como o *Gene Ontology* em Biologia Molecular (Ashburner *et al.* 2000).

3.4.4 Criação de um novo modelo de metadados *EM*

Criação de um modelo próprio do *Epidemic Marketpace*, em que seja especializado para pessoas ligadas à área da saúde. Deve ser um modelo explícito e muito bem documentado.

3.4.5 Visualização de todos os *datastreams* de um objecto digital

Todo o objecto digital é constituído por vários *datastreams*, desta forma deve ser possível visualizar todo o conteúdo de um objecto digital, através do seu identificador único. Esta funcionalidade permite compreender como é constituído um determinado objecto digital.

3.4.6 Comentar um metadado

Deve ser possível, a um utilizador, comentar expressando a sua opinião ou simplesmente solicitar alguma necessidade a outros utilizadores. Esta funcionalidade pode ser vista como um sistema de comentários semelhante ao do *Facebook*.

3.4.7 *Download* de todos os *datastreams* contidos num objecto digital

Possibilidade de se fazer o *download* de todos os *datastreams* de um objecto digital. Todos os ficheiros devem estar contidos num ficheiro compactado em formato *.zip*. Deve ser possível também especificar qual o objecto digital pretendido, através do seu identificador.

3.4.8 Ter um sistema de pedidos

Deve ser possível ter um espaço destinado para fazer pedidos onde também seja possível obter as respectivas respostas. Este sistema deve também suportar fazer *like* a um pedido.

3.4.9 Visualização de todo o conteúdo do repositório

Deve ser possível visualizar todo o conteúdo existente no repositóri, de forma intuitiva. Esta informação deve ser concisa mas perceptível a um utilizador que não tenha conhecimentos aprofundados de informática.

3.4.10 Adicionar colecções

A possibilidade de adicionar colecções ao repositório.

3.4.11 Fazer a gestão de um objecto digital

Permitir ao utilizador que adicione, altere ou elimine um objecto digital.

3.5 Requisitos dos *Web Services*

Os *web services* são responsáveis pela comunicação com:

1. Clientes - que recuperam as colecções de dados do EM e produzem gráficos dinâmicos, tendências ou mapas geográficos de acordo com a interacção do utilizador;

2. Aplicações do Epiwork - como sistemas baseados na Monitorização da *Internet* (IMS) ou plataformas computacionais (CP) para simular a propagação de doenças;

3. Outros fornecedores de dados - tais como notícias online, feeds RSS, ProMED Mail, alertas oficiais validados (OMS) e outros geradores de eventos.

Os principais requisitos dos *web services* são:

3.5.1 Implementação da capacidade de pesquisa e consulta de conjunto de dados heterogéneos

Os *web services* devem gerir o acesso a dados heterogéneos de diversas fontes, para doenças diferentes, e em diferentes formatos, através de uma *interface* de consulta ou pesquisa.

Além de informações médicas, outras, tais como geográficas, sociológicas e relacionadas com as redes de transporte, precisam de ser acedidas para simular epidemias.

A heterogeneidade dos dados depende de vários factores, tais como o seu tipo, o formato da representação e a doença. Os dados necessários para um estudo duma epidemia podem variar significativamente de doença para doença e até mesmo entre os estudos sobre o mesmo assunto, dependendo das condições experimentais e métodos de recolha de dados. Portanto, é importante definir os dados comuns e esquemas de metadados permitindo o acesso efectivo aos dados e à análise dos processos de integração da informação. Para atingir esse objectivo, a estrutura de dados de uma ampla variedade de fontes (por exemplo, IMS e CP) será caracterizada pela criação de um modelo canónico para conjuntos de dados comumente usados.

3.5.2 Implementação de uma *interface RESTful*

Os utilizadores devem ser capazes de pesquisar e consultar conjuntos de dados e correspondentes metadados através de uma *interface RESTful* [11]. Uma *interface RESTful* [11] fornece facilidade de uso, flexibilidade e simplicidade.

3.5.3 Suporte à autenticação em ambiente distribuído

Para aceder aos *web services*, os clientes devem primeiro autenticar-se, pelo menos num *site* do EM. As credenciais de autenticação devem ser partilhadas entre os *sites* do EM e qualquer *site* do EM deve aceder ao mesmo conjunto de credenciais para um dado cliente.

3.5.4 Acesso a recursos *plug-in-able*

Uma característica importante a ser suportada pelo EM, em particular para recursos de dados externos, é o acesso a recursos *plug-in-able* (Kuliberda *et al.* 2006). Estes recursos externos fornecem dados armazenados num repositório interno e podem exigir acessos virtualizados. Alguns recursos podem aparecer e desaparecer de forma inesperada, devido, por exemplo, à indisponibilidade do *site web*. Os recursos *plug-in-able* permitem a adição dinâmica de fontes de dados através de *wrappers* que garantem conexão física com uma fonte e convertem os dados recolhidos num ou mais modelos de dados canónicos suportados pelo repositório.

Capítulo 4

Implementação

"Yes, we can."(Obama, 2008).

O objectivo deste capítulo é dar a conhecer como se ligam os diferentes *softwares* e como está construído o sistema *Epidemic Marketplace*. Este capítulo está repartido por diferentes secções, começando por se explicar como o sistema está estruturado acabando com um maior detalhe para uma melhor percepção e compreensão do leitor. É dado mais ênfase na implementação dos serviços *web* visto estes terem um papel fundamental no funcionamento das várias funcionalidades que o *Drupal* oferece aos diferentes utilizadores.

4.1 Arquitectura

Com a leitura até este capítulo, já se consegue perceber que a arquitectura do sistema *Epidemic Marketplace* é composto por três grandes componentes, como ilustrado na figura 4.1:

- Gestão de Dados, onde se insere o repositório;
- SGC, Sistema de Gestão de Conteúdos, em que é disponibilizada a *interface* gráfica do repositório;
- SW, os Serviços *Web* que oferecem uma *interface* de programação para acesso ao repositório.

Como se pode verificar, na figura 4.1, toda a concepção do sistema encontra-se moldado sobre o repositório que está contido na componente "Gestão de dados".

É necessário ter muito cuidado com estes dados visto conterem informação muito sensível. Por isso, todo o possível acesso aos dados é realizado através da componente

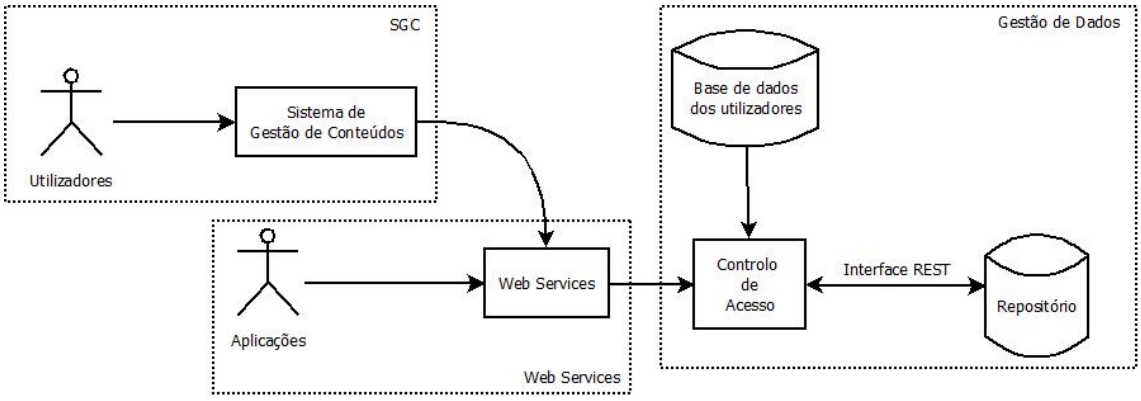

Figura 4.1: Diagrama de todo o sistema do *Epidemic Marketplace*

serviços *web*, SW. Isto é feito para que haja um maior e melhor controlo sobre os dados, o que se consegue ao restringir as funcionalidades permitidas para a gestão dos mesmos. Estes serviços *web* estão optimizados maioritariamente para as aplicações. A componente gráfica, SGC, que é usada pelos utilizadores liga-se aos serviços *web* para realizar todas as funcionalidades que o *Drupal* pode oferecer.

É feito um detalhe pormenorizado de cada componente nas secções seguintes.

4.1.1 Gestão de Dados

A componente ”Gestão de Dados”envolve a parte de armazenamento dos dados, quer epidemiológicos quer de utilizadores, e o seu controlo de acesso. A figura 4.2 exemplifica como está estruturada esta componente. Aqui torna-se importante perceber que todos os dados estão protegidos com o controlo de acesso.

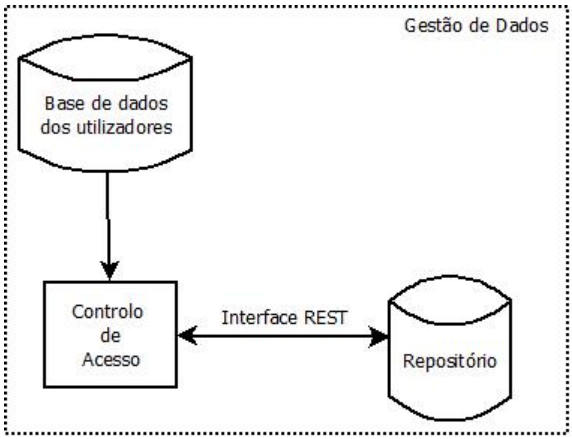

Figura 4.2: Componente Gestão de Dados

Para armazenar os dados do foro epidemiológico, foi escolhido o *Fedora Commons* versão 3.4.1, um repositório que veio de encontro às necessidades do *Epidemic Market-*

place.

Os utilizadores do *Epidemic Marketplace* sentiram necessidade de ter um outro modelo de metadados, visto conterem informações muito detalhadas e muito precisas dos vários temas abrangidos pela Epidemiologia.

O *Fedora Commons*, sendo muito flexível, permite-nos construir um novo modelo de metadados além do *Dublin Core* e armazená-lo no repositório.

A criação deste modelo de metadados foi abordado na tese de Luís Filipe Lopes, antigo aluno de Mestrado de Engenharia Informática na Faculdade de Ciências da Universidade de Lisboa. O novo modelo, *EM metadata* pode ser visualizado em [12].

Como foi referido no trabalho relacionado, capítulo 3 deste projecto, o *Fedora Commons* guarda os seus dados de forma estruturada tendo um modelo próprio para o objecto digital.

Coloca-se a questão, então como se pode guardar o novo modelo de metadados do EM no *Fedora Commons*? O modelo do objecto do *Fedora Commons* contém, na perspectiva do objecto digital, uma secção própria para guardar o novo modelo de metadados, o *datastream*. Apesar deste *datastream* ser visto pelo *Fedora Commons* como opcional, todo o sistema do EM foi concebido para ”olhar”apenas para este *EM metadata*. Ou seja, qualquer operação que ocorra no repositório é apresentada a informação contida no *datastream EM*. Desta forma todos os objectos que sejam inseridos no repositório terão que ter o metadado *EM (EM metadata)* sendo só assim possível fazer a integração do novo modelo de metadados no *Fedora Commons*. O *DC* apenas contém a informação necessária para o *FC* poder trabalhar, tais como o *PID* (Identificador Único) e o título do objecto.

O *Fedora Commons* é um *software* ”cego”, ou seja, armazena os seus dados sem perceber como estes estão organizados. Desta forma e como referido no trabalho relacionado, o *FC* precisa de um *datastream* especial, o *RELS-EXT*, que lhe diga como estão organizados e relacionados entre si os objectos no seu repositório.

A melhor forma de estruturar os vários objectos do *Fedora Commons*, com informação variada, é pensar na estrutura de um sistema de ficheiros *Windows* ou *Linux*. Ou seja, os dados serem estruturados em colecções que podem ser vistas como pastas (no sistema de ficheiros). A única informação necessária a guardar, para concretizar este sistema de colecções, é:

- o identificador do ”Pai”, *PID*. Desta forma consegue-se perceber em que ”pasta”está inserido um dado objecto;
- a categoria do ”Pai”, *type*. Desta forma, consegue-se perceber em que ”tema”se encontra um dado objecto.

Pode ser visualizado um exemplo, muito simples, na figura 4.3 em que é apresentado uma possível estrutura de um repositório. Vamos supor que este repositório contém dados

sobre jogos de tabuleiro.

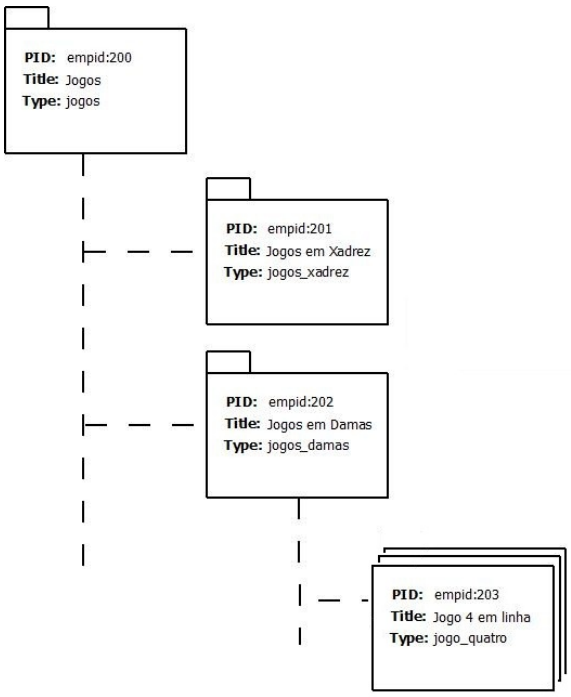

Figura 4.3: Explicação das colecções

A figura 4.3 indica que existem, pelo menos, quatro objectos digitais no repositório. Estes estão divididos por categorias, onde existe a colecção principal ”jogos”. Dentro desta colecção encontra-se então os jogos de tabuleiro, xadrez e damas, em que se pode considerar que o jogo 4 em linha tem a mesma estrutura que o jogo das damas e por isso está dentro da colecção damas.

Com este sistema de organização é possível tirar o máximo partido da modelação de objectos digitais no *Fedora Commons*.

Um exemplo deste *RELS-EXT* pode ser visualizado na figura 4.4. Em que nos indica que o objecto *empid:473* (1) é uma colecção (2) e está inserido na ”pasta”islandora:top (3) que é considerada como sendo a ”pasta”raiz, principal.

```
▼<rdf:RDF xmlns:fedora-model="info:fedora/fedora-system:def/model#" xmlns:rdf="http://w
 system:def/relations-external#">
  ▼<rdf:Description rdf:about="info:fedora/empid:473"> 1
    <rel:isCollection>true</rel:isCollection> 2
    <rel:isMemberOf rdf:resource="info:fedora/islandora:top"/>3
    <fedora-model:hasModel rdf:resource="info:fedora/islandora:collectionCModel"/>
   </rdf:Description>
 </rdf:RDF>
```

Figura 4.4: *Datastream RELS-EXT*

Uma outra funcionalidade que o *EM* oferece é a possibilidade de se poder comentar num dado objecto digital. Por exemplo, querer informar um utilizador, especializado numa área, que recolheu novos dados sobre um caso de estudo em causa. Esta funcionalidade pode ser vista como o sistema de comentários do *Facebook*. Esta particularidade foi implementada seguindo o mesmo raciocínio que o armazenamento do *EM metadata*. Ou seja, foi adicionado ao modelo do objecto digital um novo *datastream* que neste caso se chama de *Annotation*. Este *datastream* contém a informação sobre a data do comentário, *username* que fez o comentário, a *flag like* que informa se um utilizador colocou *like* no comentário, identificador desse comentário e o próprio texto do comentário.

Na figura 4.5 é possível visualizar um exemplo de um ficheiro *Annotation.xml*.

```
▼<Annotations>
   <Annotation date="2012/1/7" username="fedoraAdmin" like="0" id="0">I need more details about this resource.</Annotation>
 </Annotations>
```

Figura 4.5: Exemplo de um objecto digital que contém um comentário

A informação referente ao utilizador também se reveste de muita importância.

O *Epidemic Marketplace* prepara-se para ter vários utilizadores, deste modo é necessário garantir e controlar o seu acesso. A melhor forma de fazer essa gestão de utilizadores é implementar uma base de dados que contenha toda a informação referente aos diversos utilizadores. A base de dados é o *OpenLDAP*.

O *OpenLDAP* tem a particularidade de poder guardar, além da informação do utilizador, a informação do papel (*role*) atribuído a esse utilizador. Um possível papel (*role*) pode ser o de ”leitura”, onde o utilizador tem apenas acesso à leitura de todos os objectos sem que os possa editar.

A estrutura da informação no *OpenLDAP* é conseguida através de *schemas*, que são tipos de dados que ajudam a definir a estrutura da informação. Por isso, instalou-se o *schema eduPerson*. Este *schema* contém um atributo próprio para guardar a informação do utilizador, o *eduPerson Afilliation*.

O *eduPerson Afilliation* pode conter vários valores o que possibilita atribuir mais do que um papel ao utilizador, por exemplo, um utilizador ter o papel de leitura sobre todos os objectos e o de Administrador de uma colecção em particular. A restante informação relativa ao utilizador é guardada noutro *schema* definido pelo *slapd*, como o *e-mail*, página pessoal, etc.

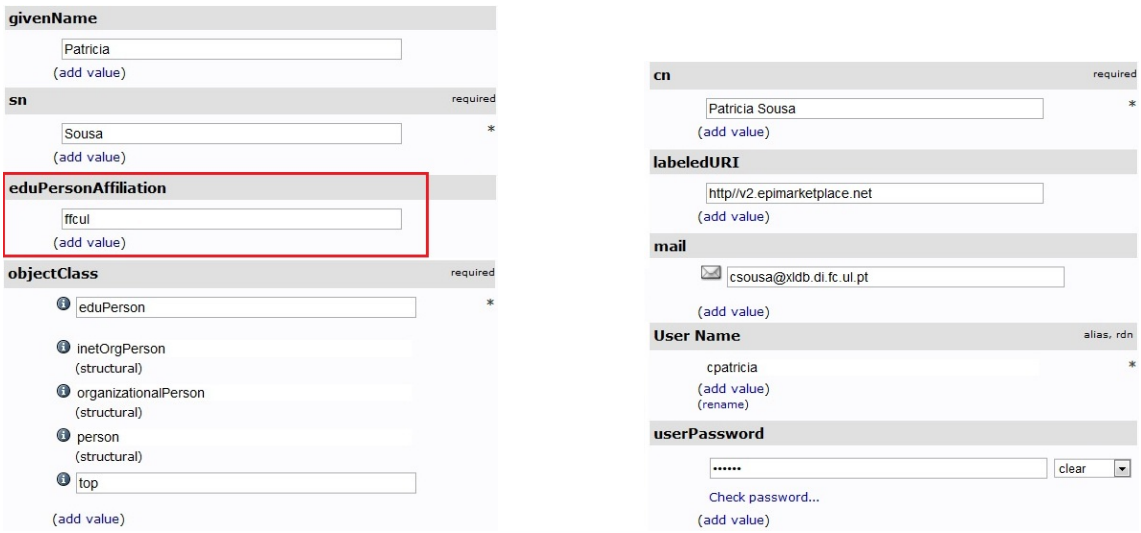

Figura 4.6: Exemplo de como a informação se encontra no *OpenLDAP*

A figura 4.6 apresenta uma entrada na base de dados *OpenLDAP*, neste caso é informação referente ao meu utilizador. A variável *eduPersonAffiliation* (do lado esquerdo da imagem) contém o valor da *FFCUL*, em que previamente se definiu que este papel (*role*) iria estar associado Ua gestão de todo o sistema. Caso se pretendesse adicionar mais que um papel, por exemplo, de administrador, iriamos ter na mesma variável dois valores. Um referente ao valor *FFCUL* e outro de administrador. A informação que aparece do lado direito da imagem refere-se exclusivamente à informação do utilizador.

A autenticação do utilizador é realizada com uma pesquisa no directório do *openLDAP* e é um processo simples. Todo o processo de *login* pode ser acompanhado na figura 4.7. É primeiro usado um utilizador ”administrator”, *OpenLDAP Admin*, que realiza uma pesquisa no directório com base no *distinguished Name (DN)* do utilizador, que neste caso será cpatricia. É feita uma comparação da entrada encontrada no directório e a entrada com que se está a trabalhar. Caso sejam iguais é enviada uma resposta em como o utilizador que se quer autenticar é fidedigno. Caso contrário é negado o acesso.

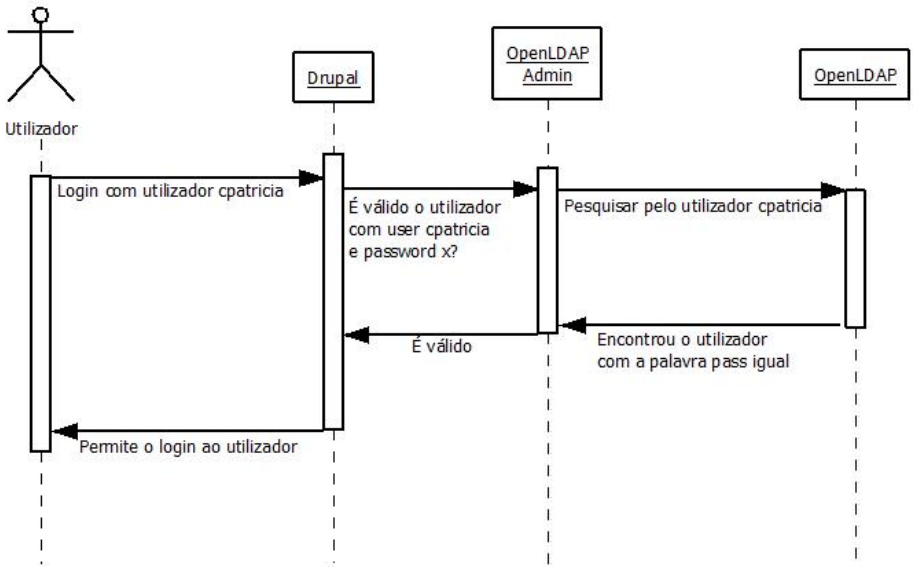

Figura 4.7: Sequência de como se processa o *login*

Resta perceber como funciona o acesso aos diferentes objectos existentes no repositório e para isso coloca-se a questão, de que forma se pode definir que tipo de acesso um dado objecto deve ter? A resposta a essa questão é dada através de políticas *XACML* que estão ”acopladas”nos diferentes objectos digitais do *FC*. O *FC* na versão 3.4 permite guardar estas políticas juntamente com os objectos digitais.

De toda a concepção do controlo de acesso está encarregue o João Zamite, aluno de Doutoramento em Engenharia Informática na Faculdade de Ciências da Universidade de Lisboa.

No entanto, o processo de autenticação aos diferentes recursos é realizado da seguinte forma (que pode ser acompanhado pela figura 4.8):

É feito um pedido para obter um determinado objecto aos serviços *web*. Este pedido é entregue ao *Fesl* que vai primeiro verificar se o utilizador pertence à comunidade *EM* e para isso vai ao *OpenLDAP* procurar a informação necessária, que neste caso são os atributos e o grupo ao qual pertence.

Após a obtenção desses dados o *Fesl* encarrega-se de ir ao *Fedora Commons* procurar o *XACML* pertencente ao objecto requisitado. Com estes dados o *Fesl* valida se o utilizador tem direito, através do seu papel, ao objecto em si. Se sim, então é realizado um novo pedido ao *FC* para ir buscar o objecto digital inicial.

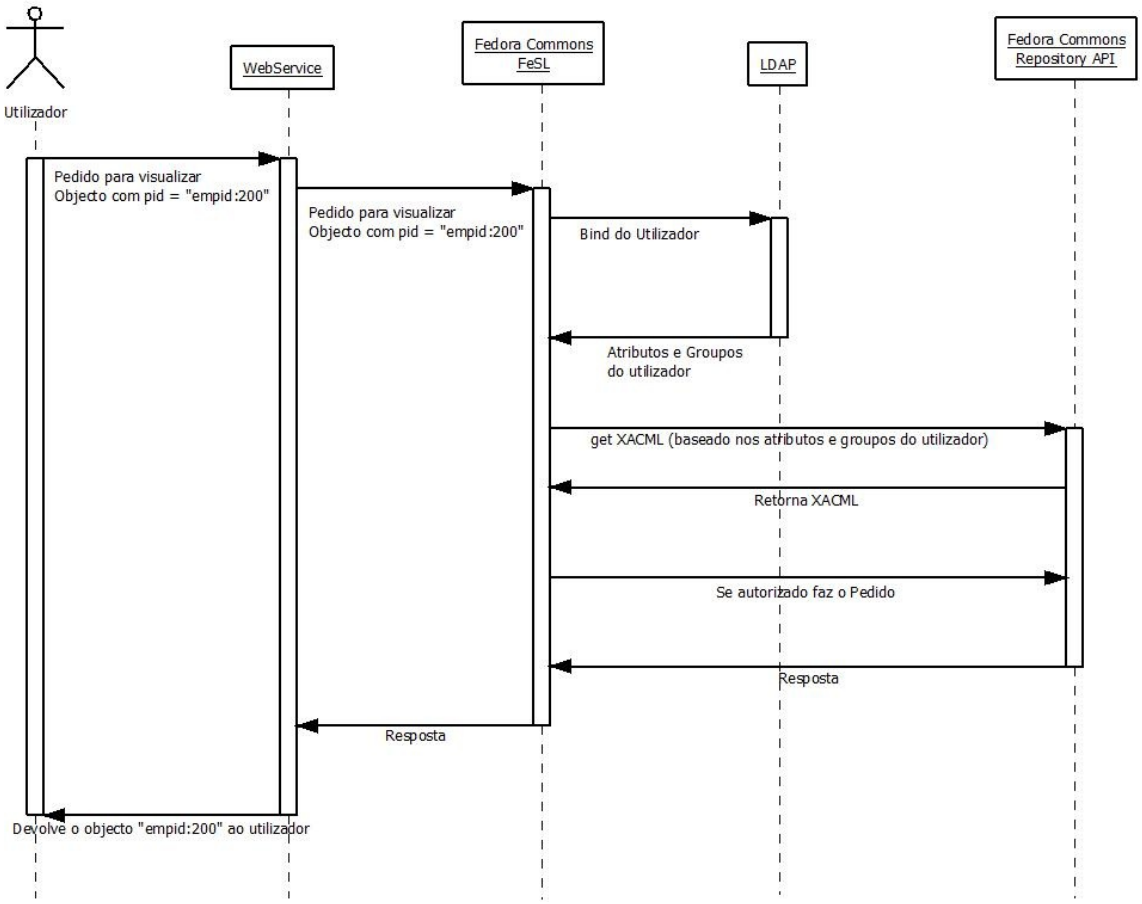

Figura 4.8: Autorização nos objectos digitais

4.1.2 Serviços *Web*

Os serviços *web* têm um papel fundamental visto serem o intermediário entre o repositório e as aplicações e/ou os utilizadores. A versão 1 contou com quatro serviços *web*, que foram implementados por Hugo Ferreira, antigo aluno de Mestrado de Engenharia Informática na Faculdade de Ciências da Universidade de Lisboa [21]. Os seguintes *web services* podem ser explicados em melhor detalhe na tese de Hugo Ferreira:

- Search objects
- Search collections
- Fetch
- Upload

A versão 1 contribuiu bastante para a evolução e personalização dos novos serviços *web*, neste momento foi possível afiná-los tendo em conta o feedback da primeira versão do *EM*.

Esta secção irá repartir-se por funcionalidades partindo do melhoramento dos serviços *web* antigos prosseguindo até aos novos *web services*.

A nova versão dos Serviços *Web* contém mais serviços, nomeadamente:

- Serviço Web de Pesquisa sobre os dados do repositório - *search*
- *Download* de um objecto digital - *Fetch*
- *Upload*
- Realizar *Download* de todos os *datastreams* de um objecto digital - *rawfetch*
- Apresentar a estrutura do respositório - *repositoryTree*
- Visualização de todos os comentários realizados no sistema - *listAnnotations*
- Sistema de Pedidos - *request*
- Criar novas colecções no repositório - *category*
- *List Datastreams*
- *List Results of a term*
- *Update* ou *Delete Dataset*

Serviço Web de Pesquisa sobre os dados do repositório - *search*

Esta funcionalidade foi readaptada e melhorada da versão 1.

O problema que existia da versão anterior era o facto de a pesquisa insidir sobre os metadados *Dublin Core*. O serviço de pesquisa usava o *search* do próprio *Fedora Commons* ”mascarado”para uma melhor compreensão do serviço.

Inicialmente não era problemático a pesquisa não incidir sobre o novo modelo *EM*, visto este ainda estar em fase de desenvolvimento. Para a segunda fase era fundamental e imprescindível que a pesquisa fosse alterada para albergar o novo modelo do *EM*.

Para concretizar esta funcionalidade passou por se instalar o *software Solr*. Este guarda todo o conteúdo de todos os *datastreams* do objecto digital. O *Solr* guarda este conteúdo como atributo/valor, um género de base de dados. Por exemplo, consideremos

Atributo	Valor
em.title	CDCTravelNotices_-_2011 02 15_10:38_53Data
em.subject	epidemic
em.generalDescription.description	Messages from the CDC Travel Notices Rss Feed containing the name of the disease and location
em.generalDescription.format	format
em.generalDescription.language	en - English
em.generalDescription.type	dataset
em.generalDescription.URL	http://
em.date	2011-02-15
em.dateSubmitted	2011-02-15

Tabela 4.1: Como o *Solr* vê os dados no Repositório

um ficheiro que contém o novo modelo EM (fig. 4.9). Este modelo contém várias *tags*, *em:title* por exemplo, que posteriormente são mapeadas e guardadas como mostra a tabela 4.1.

```
▼<em:em xmlns:em="http://epiwork.di.fc.ul.pt/metadata/" xmlns:xlink="http://www.w3.org/1999/xlink" x
   <em:title>CDCTravelNotices_-_2011-02-15_10:38:53Data</em:title>
   <em:subject>epidemic</em:subject>
  ▼<em:generalDescription>
    ▼<em:description>
       Messages from the CDC Travel Notices RSS Feed containing the name of a disease and a location
     </em:description>
     <em:DOI/>
     <em:format>format</em:format>
     <em:language>en - English</em:language>
     <em:type>dataset</em:type>
     <em:URL>http://</em:URL>
     <em:version/>
   </em:generalDescription>
   <em:date>2011-02-15</em:date>
   <em:dateSubmitted>2011-02-15</em:dateSubmitted>
```

Figura 4.9: *EM Metadata*

O search da versão 2 irá então à ”tabela”do *Solr* pesquisar pelo atributo. Por exemplo, queremos pesquisar pelo título ”epidemiologia”. O pedido é feito ao serviço *web* que o transforma na linguagem do *Solr* e o envia para o mesmo. A resposta do *Solr* é um *array* de *PIDs* que, neste caso, conterá todos os objectos que tenham como título ”epidemiologia”. Por fim, esta resposta é recebida pelo *web service search*, que posteriormente irá para cada objecto, do *array*, obter o seu conteúdo para o mostrar ao utilizador. Este procedimento pode ser visto na sequência representada na figura 4.10.

O preenchimento da ”tabela”do *Solr* é feito por intermédio de um serviço oferecido pelo *Fedora Commons*, chamado de *UpdateIndex*. Este serviço, que é chamado por meio de uma *interface RESTful* tem que ser induzido por meio de um outro serviço *web* privado do *EM*. Este serviço *web* privado, apenas realiza o preenchimento da ”tabela”do *Solr* e é despoletado quando é inserido, removido ou alterado um objecto digital no repositório.

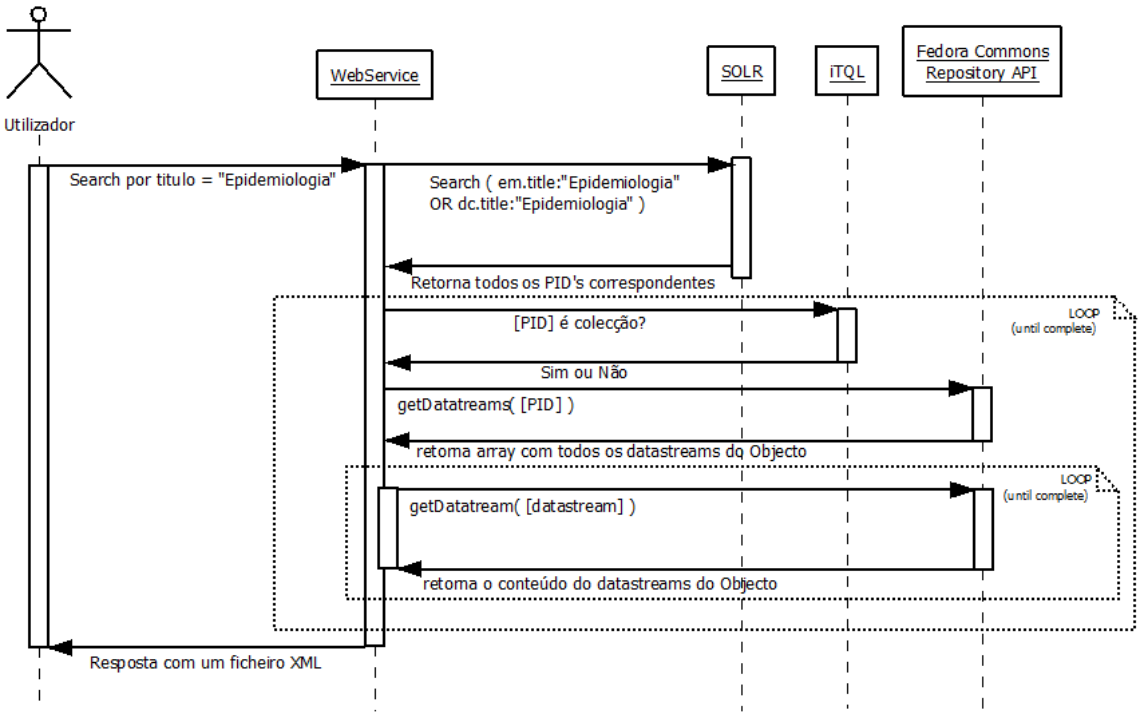

Figura 4.10: Diagrama de Sequência que realiza a pesquisa no repositório

O serviço *UpdateIndex*, que é chamado pelo serviço *web* privado, utiliza uma *URL* fixa onde é indicado qual o objecto que se pretende ”introduzir na tabela”, através do seu *PID*. A *URL* é a seguinte:

```
http://localhost:8443/fedoragsearch/rest?operation=updateIndex
&action=from Pid&value=[PID]
```

em que o *value* (valor) corresponde ao *PID* do objecto digital.

Para fazer a remoção de um valor na ”tabela”do *Solr* é simplesmente actualizar o *Indice* do objecto e informar o *FC* que queremos eliminar este mesmo objecto. Para tal utiliza-se a *URL*,

```
http://localhost:8443/fedoragsearch/rest?operation=updateIndex
&action=deletePid&value=[PID]
```

mais uma vez, em que o *value* (valor) corresponde ao *PID* do objecto digital.

O serviço *web search* conta com duas possibilidades, uma de podermos realizar pesquisa sobre todos os dados do reposiório e outra só perante colecções. De referir que toda a documentação dos serviços *web* se encontra em `http://v2.epimarketplace.net/developers_corner/`.

Caso se pretenda pesquisar sobre todos os objectos existentes no repositório, basta utilizar a seguinte *URL*:

```
http://api.epimarketplace.net/search/objects/
[Parameter key]/[Parameter value]
```

onde o *Parameter Key* pode ser *title*, *subject*, *creator*, etc.

Para se perceber melhor como funciona podemos simular o mesmo caso que a sequência da figura 4.10, no entanto, ao invés de se pesquisar pela palavra *epidemiology* pesquisar-se por *epidemic*. Isto porque o objecto retornado contém um metadado *EM* menor, tornando-se mais fácil de perceber qual a estrutura do ficheiro *XML* retornado pelo serviço *web Search*.

Portanto, o que queremos perceber é quantos objectos digitais contém no seu título a palavra *epidemic* e para isso foi introduzida a seguinte *URL*,

```
http://api.epimarketplace.net/search/objects/title/epidemic
```

Esta pesquisa retornou um ficheiro *XML* com vários objectos digitais mas por agora é apresentado um resultado e pode ser visualizado na figura 4.11.

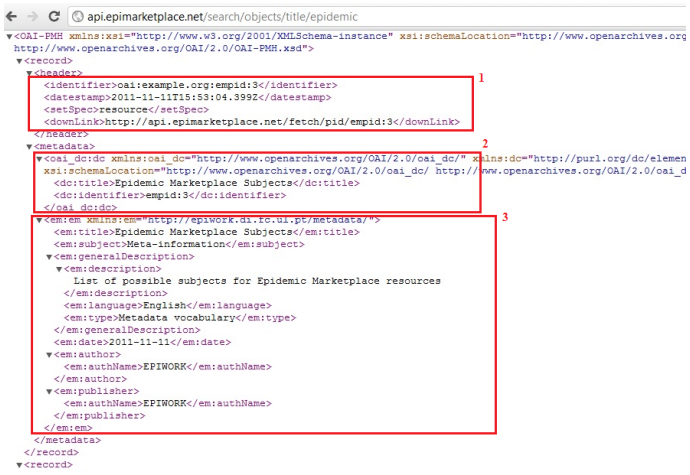

Figura 4.11: Pesquisa no repositório por objectos que contenham *epidemic* no seu título

Este ficheiro *XML* está de acordo com o protocolo *OAI-PMH* e com isto é possível com que outros repositórios possam conter os nossos objectos digitais. O protocolo requer que o ficheiro *XML* contenha um cabeçalho (*header*) e um espaço para os metadados (*metadata*) quer *DC* ou *EM*. Como se pode verificar temos um cabeçalho (1) em que nos informa qual o identificador do objecto, a data em que foi submetido, se é um recurso ou colecção e a possibilidade de se poder fazer o *download* do objecto. Este *download* nada mais é do que a prestação de outro serviço *web*, *fetch*, que irá ser detalhado na subsecção seguinte.

Por fim, a *tag metadata* contém todos os metadados existentes no objecto. Neste caso temos os metadados *DC* (2) e *EM* (3).

Podemos querer uma pesquisa mais refinada, por exemplo, querer saber se existe no repositório objectos que contenham como autor *FFCUL* e como título *collector* e para isso utiliza-se a seguinte *URL*:

```
http://api.epimarketplace.net/search/objects/
author/FFCUL/title/collector
```

É interessante conjugar várias possibilidades para a nossa pesquisa com um leque variado de *Parameter Key*.

No caso de se querer apenas pesquisar em colecções, basta alterar um pormenor na *URL*, que em vez de utilizar a palavra *objects* utiliza-se a palavra *collection*. Ou seja,

```
http://api.epimarketplace.net/search/collections/
[Parameter Key]/[Parameter Value]
```

Download de um objecto digital - *Fetch*

O serviço *fetch* foi igualmente transformado para esta nova versão do *EM*. Este serviço tem como finalidade devolver todo o conteúdo do objecto digital que se encontra no repositório. Por intermédio do *PID* do objecto é devolvido um ficheiro *XML* que contém um cabeçalho (1) com título, identificador do objecto (*PID*) e o seu tipo (colecção ou recurso) e por fim todos os *datastreams* deste objecto, incluindo um *datastream* especial que pode conter qualquer tipo de ficheiros (2). Este ficheiro, retornado pelo serviço *web fetch*, pode ser visualizado na figura 4.12.

Suponhamos que o utilizador pretende guardar no repositório um ficheiro txt, por exemplo, que contém dados sobre epidemiologia. Para que este ficheiro seja guardado, tem que ser transformado em *base64*. Assim garantimos que o ficheiro permanece no repositório por tempo indeterminado e de forma coerente.

A grande diferença da versão 1 para a 2 é a rapidez no serviço. A versão 2 contou com a ajuda da linguagem *iTQL*, esta serve para inquirir ficheiros *RDF/XML*, útil para pesquisar no *datastream RELS-EXT*. Isto porque é onde se encontra toda a informação referente às ligações de um objecto aos restantes.

Neste caso, usou-se a linguagem *iTQL* para se perceber se o objecto pedido é colecção ou não, para construir a *tag collection* ou *resource*. A linguagem *iTQL* tornou-se numa

Figura 4.12: Ficheiro XML que contém a resposta do *web service fetch*

ferramenta poderosa, tornando a pesquisa no repositório muito mais rápida do que a anterior versão do serviço *web fetch*. As *queries* baseiam-se no formato sujeito, predicado e valor, como explicado no trabalho relaccionado.

Abaixo encontra-se a *query* utilizada para perceber se um dado objecto é colecção ou não.

```
Select $object $identifier $var from <#ri> [1]
where ( $object <dc:identifier> $identifier [2]
and $identifier <mulgara:is> 'empid:200' [3]
and $object
<info:fedora/fedora-system:def/
relations-external#isCollection $var [4]
and $object
<fedora-model:state> <info:fedora/fedora-system:def/
model#Active> ) [5]
```

Como se pode verificar esta *query* devolve um objecto, o seu identificador e uma variável que é utilizada na claúsula *where* e que devolve *true* ou *false* [1]. A *query* começa por indicar que o objecto tem o identificador que o utilizador pediu, neste caso é o *empid:200* [3]. É curioso verificar que na linha [2] não se colocou a referência *em: identifier* e sim *dc:identifier*. Isto acontece devido ao facto de o *Fedora Commons* guardar, para todo o objecto digital o metadado *Dublin Core*, que contém as *tags* título e identificador. Pode dar-se o caso de o utilizador, inconscientemente, eliminar o identificador do *EM metadata*, nessa situação torna-se perigoso fazer uma *query* sobre um predicado que pode ser nulo.

A linha [4] é que vai fornecer a informação se um dado objecto é colecção ou não. A variável *var* contém os valores de *True* ou *False*. Caso seja *True* é inserido no *XML* final a *tag collection*. Caso contrário é colocada a *tag resource*.

Para se utilizar este serviço basta utilizar a seguinte *URL*:

```
http://api.epimarketplace.net/fetch/pid/[value]
```

Upload

Outro serviço que a versão 1 oferecia era o *upload*. Este serviço é muito especial, visto ter sofrido grandes alterações.

O serviço funciona através do método *POST* e de um ficheiro de entrada em formato *OAI-ORE*. A documentação deste serviço encontra-se em `http://v2.epimarketplace.net/developers_corner/web_services/upload_object`.

A versão 1 apenas suportava a inserção de um ficheiro de cada vez e em formato *DC*.

Esta nova versão, vem rica em alterações. Agora é possível fazer várias inserções de metadados diferentes, que tanto podem ser em *DC* como em *EM metadata*.

Neste caso o protocolo *OAI-ORE* não permite que se utilize outros formatos que não o *DC*, no entanto, o serviço *upload* permite que se use o formato *EM*.

O serviço, antes de inserir um objecto digital no repositório, verifica se este contém um título, visto ser uma obrigatoriedade do *FC*. Após isso é verificado se este objecto já contém um identificador, caso contenha é substituído pelo seu novo identificador. Todo o mecanismo encontra-se descrito na figura 4.13 do lado direito. Uma das particularidades deste serviço é o facto de este permitir adicionar em diferentes colecções vários ficheiros. Na versão 1, apenas se fazia a inserção de um ficheiro numa colecção. Para indicar ao sistema que se pretende inserir um ficheiro, por exemplo, um ficheiro .txt que contém dados recolhidos sobre o H1N1, o utilizador deve colocar na *tag* em:description o atributo *base64* a *True*, ou seja,

```
<em:description base64="True">
```

Outra das particularidades desta nova versão é o de dar ao utilizador a liberdade de ter várias *tags* iguais mas com conteúdos diferentes, por exemplo, um objecto digital contém dois *subjects*. Onde um indica que se trata de um caso de gripe (*Flu*) e outro especificando que é um caso de *H1N1*. Para isso utiliza as seguintes *tags*,

```
<em:subject>Flu</em:subject>
<em:subject>H1N1</em:subject>
```

Aqui é possível verificar que existem dois ”blocos”iguais. Com esta nova versão é, então, possível construir vários ”blocos”com a mesma *tag* em diferentes *tags*.

Portanto, para se poder utilizar este serviço, é necessário construir o nosso ficheiro *OAI-ORE* e para isso é preciso saber primeiro em que colecção desejamos colocar o nosso

objecto. No exemplo da figura 4.13 (do lado esquerdo) estamos a indicar que queremos colocar o objecto digital, descrito em (2) na colecção empid:330 (1).

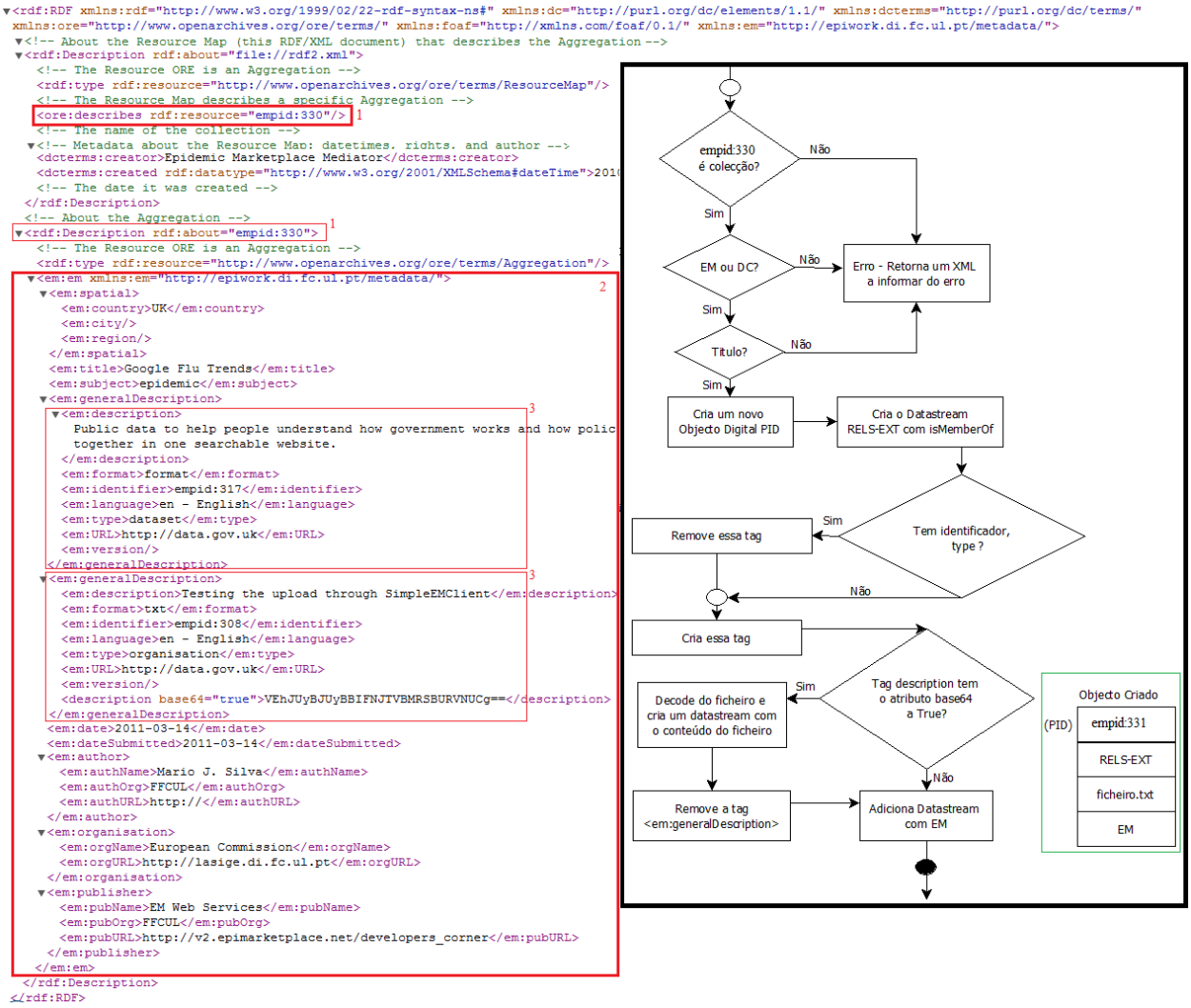

Figura 4.13: Documento *OAI-ORE* a ser enviado para o serviço *web upload*

Na figura 4.13 é também possível verificar que existem dois ”blocos”em:generalDescription (3), onde um deles indica uma descrição normal e outro indica um ficheiro .txt a ser introduzido juntamente com o objecto digital. Neste último caso é especificado na *tag description* o atributo *base64*, desta forma o sistema compreende tratar-se de um ficheiro e procede ao seu *decode*. Posteriormente é criado um novo *datastream* com o nome do ficheiro mais a sua extensão e eliminado do metadado *EM* a sua referência, isto para não aparecer no metadado informação própria para o entendimento da plataforma *EM*.

Na parte direita da figura 4.13 compreende-se todo o processamento realizado por parte do serviço *web upload*. Começa por perceber se o objecto indicado em (1) é de facto uma colecção, isto para precaver qualquer possibilidade de incoerências no repositório.

Após esta validação é então validado se o novo objecto digital contém a *tag* em:title. Este é um ponto fundamental para se poder criar um novo objecto digital no *Fedora Commons*, só com essa informação é possível criar novos objectos digitais. Reunida esta informação pode-se então criar o *datastream RELS-EXT* onde é esclarecido que o novo objecto digital se encontra dentro da colecção empid:330 e para isso basta colocar o seguinte, no *datastream*,

```
<rel:isMemberOf rdf:resource="info:fedora/empid:330">
</rel:isMemberOf>
```

Após a criação deste *datastream* é validado se o novo objecto contém as *tags identifier* ou *type*, caso tenha são removidas para conter o valor do seu novo identificador e da categoria do seu ”pai”, respectivamente.

O passo seguinte é tentar perceber se a *tag generalDescription* contém algum ficheiro e proceder ao seu *decode*, criação de um *datastream* com o nome do ficheiro e remoção no metadado *EM*.

Por fim pode-se criar o *datastream EM* no objecto digital novo, com *PID* empid:331 e pode ser ”visualizado”na imagem 4.13 no canto inferior direito.

Realizar *Download* **de todos os** *datastreams* **de um objecto digital -** *rawfetch*

Este serviço *web* tem duas particularidades uma é a possibilidade dada ao utilizador de realizar o *download* de todos os *datastreams* de um objecto digital, outra é a de poder apenas referir qual o *datastream* do objecto digital que o utilizador pretende visualizar. Estas duas funcionalidades utilizam dois métodos disponíveis no *Fedora Commons*, apenas se pretende dar ao utilizador uma forma mais intuitiva de os chamar além de estes não estarem disponíveis externamente. Grande parte devido ao facto de o repositório não ter um *IP* externo e o utilizador não ter conhecimento dos métodos do *Fedora Commons*.

Iremos abordar a primeira particularidade que será o facto de o utilizador poder realizar o *download* de todos os *datastreams* de um objecto digital. Este serviço *web* é muito simples e temos a figura 4.14 para ajudar na compreensão do raciocínio.

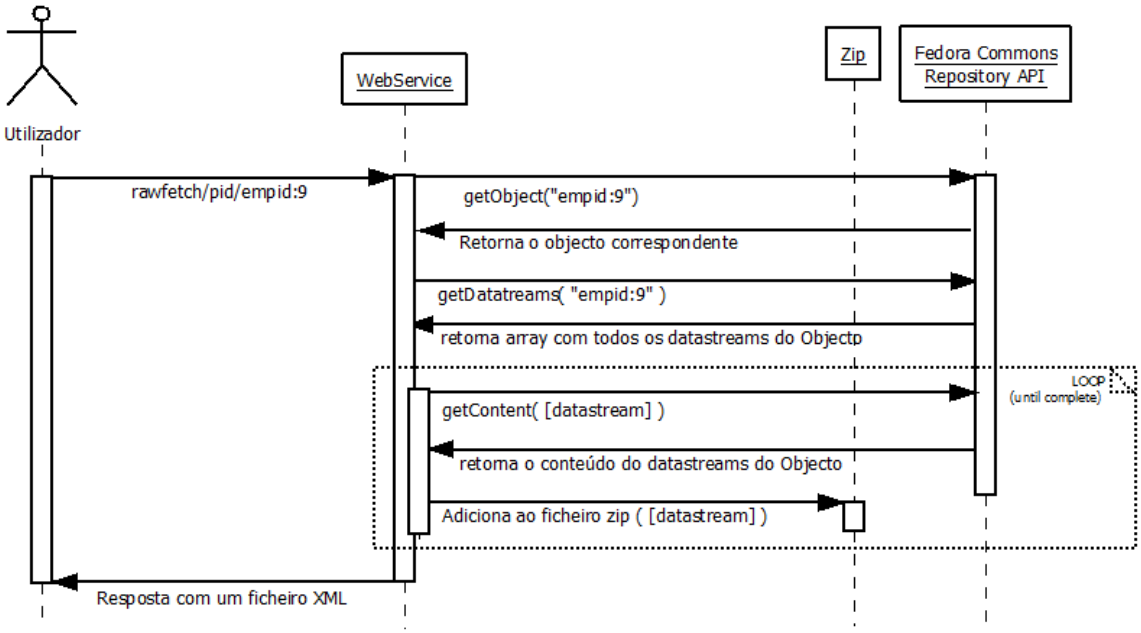

Figura 4.14: *Download* de todos os *datastreams* de um objecto digital

O serviço começa por obter o objecto digital que neste caso é o empid:9. Após obter a instância desse objecto vai obter todos os seus *datastreams* por intermédio do método *getDatastreams* do *Fedora Commons*. Para cada *datastream* o serviço *web* vai colocar num ficheiro *.zip*. Assim dá a possibilidade ao utilizador de ter todos os *datastreams* guardados num ficheiro só.

Este serviço *web* é obtido com a seguinte *URL* e com o método *GET* do protocolo *HTTP*:

```
http://api.epimarketplace.net/rawfetch/pid/empid:9
```

Caso o utilizador tenha curiosidade em visualizar apenas o *datastream RELS-EXT* para, por exemplo, perceber a sua relação com os outros objectos digitais, pode introduzir na *URL* mais dois parâmetros, são eles:

```
http://api.epimarketplace.net/rawfetch/pid/empid:9/
datastream/RELS-EXT
```

Este serviço *web* é semelhante ao de cima, com o detalhe de se especificar qual o *datastream* em causa. Este serviço é, sem dúvida, um simples chamar de uma função do *Fedora Commons* que neste caso é *getDatastreamDissemination*, como se pode comprovar na figura 4.15.

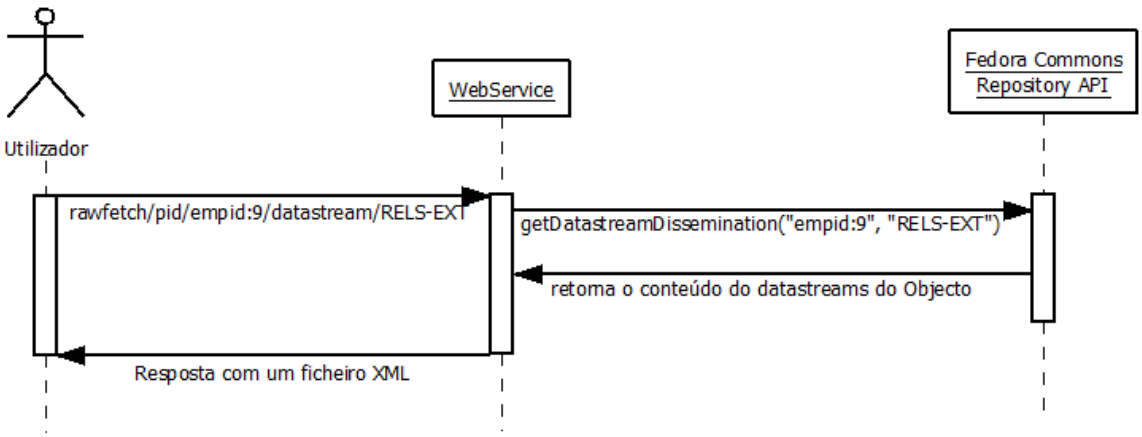

Figura 4.15: Serviço *web* em que se visualiza apenas um *datastream*

Apresentar a estrutura do respositório - *repositoryTree*

Este serviço *web* foi desenvolvido porque se sentiu a necessidade de perceber como se encontrava a estrutura dos objectos digitais no repositório. A estrutura do repositório está construída à semelhança de um sistema de ficheiros, *Linux* ou *Windows*. Por essa mesma razão a implementaçao deste serviço teve por base o princípio fundamental do sistema de colecções, ou seja, apenas e somente as colecções têm ”filhos”. Qualquer objecto digital que não tenha relação com outros, não ter uma característica comum, é considerado um ”filho”.

O desenvolvimento deste serviço tirou partido das funções recursivas, onde para cada objecto vai perceber se este tem ”filhos”, caso tenha vai então perguntar a cada um dos ”filhos”se é colecção ou não. Em caso afirmativo é então colocada, na construção do *XML*, a *tag* collection, caso contrário é colocada a *tag resource*. A figura 4.16 apresenta na íntegra todo o processamento existente neste serviço.

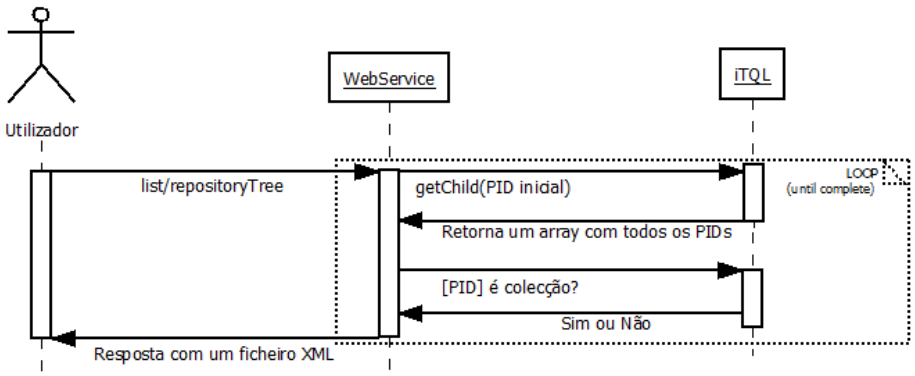

Figura 4.16: Visualizar o conteúdo do repositório

Para se perceber melhor qual a estrutura do ficheiro que é retornado ao utilizador apresenta-se uma ilustração (figura 4.17) do ficheiro *XML*

Figura 4.17: Imagem do ficheiro *XML* que é retornado no serviço *web, repositoryTree*

O ficheiro que é retornado contém uma estrutura muito simples, uma vez que não se pretende dar muita informação ao utilizador, apenas a essencial. O ficheiro, neste caso, contém uma colecção com um cabeçalho (1) em que nos indica que se trata do *PID empid:1* e tem o nome de *emresources* seguido dos seus *links* para pesquisa e *download*. Esta colecção contém pelo menos um objecto do tipo (*resource*) com o título *Arrival Times Global* e com *PID* empid:43. Todos os recursos que estejam dentro de uma colecção estão sempre contidos na *tag content* (2). É possível verificar que a informação que é apresentada ao utilizador não é muita mas é essencial para perceber toda a estrutura do repositório.

Este serviço é acedido através do *URL* com o método *GET*:

```
http://api.epimarketplace.net/list/repositorytree
```

É interessante que o *Epidemic Marketplace* também oferece ao utilizador a possibilidade de definir o nivel de profundidade com que se pretende visualizar o conteúdo do repositório. Vamos supor que o utilizador apenas prentende perceber quantas ”pastas”e recursos se encontram no primeiro nível, para isso basta adicionar à *URL* os seguintes parâmetros:

```
http://api.epimarketplace.net/list/repositorytree/
level/1
```

Aqui é possível perceber, por intermédio das *tags collection* ou *resource*, quantos recursos ou colecções estão no primeiro nível.

Vamos supor que o utilizador, mesmo assim, pretende perceber quantas ”pastas”ou recursos estão no objecto digital empid:1 no segundo nível. Ou seja, apenas iremos visualizar os ”filhos”do empid:1, uma vez que o primeiro nível é o próprio objecto empid:1 e o segundo nível corresponde ao dos seus ”filhos”. Portanto a *URL* a ser utilizada é:

```
http://api.epimarketplace.net/list/repositorytree/
pid/empid:1/level/2
```

Visualização de todos os comentários realizados no sistema - *listAnnotations*

Um dos requisitos impostos ao *Epidemic Marketplace* seria a construção de um sistema de comentários. Um sistema de comentários em que os utilizadores podiam utilizar para pedirem ajuda sobre o tema de um objecto digital ou simplesmente contactar o ”dono”do objecto digital para o informar que tinha consigo um documento que poderia juntar ao seu objecto digital. É neste sentido que surge este serviço *web*, onde é possível apresentar todos os comentários existentes no repositório, adicionar novos comentários ou até mesmo eliminar comentários.

Para se perceber melhor o funcionamento deste serviço *web* temos a tabela 4.2 que contém o método *HTTP* a utilizar, a *URL* utilizada e por fim uma breve descrição do serviço:

Os comentários encontram-se no *datastream Annotation* de cada objecto digital. Desta forma o serviço *web* apenas ”pergunta”a todos os objectos se tem o *datastream Annotation*, caso tenha então obtém o seu conteúdo e apresenta-o de forma estruturada ao utilizador. A estrutura é muito simples, apenas é apresentado ao utilizador o conteúdo do *datastream Annotation* de cada objecto, como se pode visualizar na figura 4.18.

Método	URL	Descrição
GET	http://api.epimarketplace.net/annotation/list	Permite a visualização de todos os comentários do repositório
POST	http://api.epimarketplace.net/annotation	Permite adicionar novos comentários ao repositório
PUT	http://api.epimarketplace.net/annotation	Permite alterar o estado do comentário, ou seja o *Like*

Tabela 4.2: Métodos utilizados para a utilização do serviço *web listAnnotations*

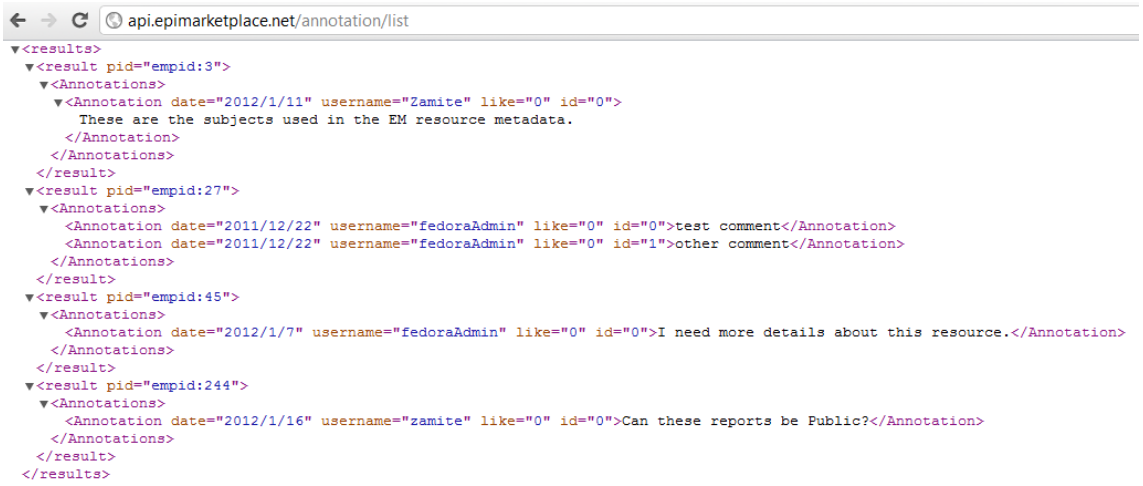

Figura 4.18: Visualização de todos os comentários existentes no repositório

Aqui é possível verificar que, pelo menos, quatro objectos digitais têm comentários, sendo que o objecto digital com *PID* empid:27 tem dois comentários.

Os métodos *POST* e *PUT*, da tabela 4.2, requerem dois ficheiros de entrada. No primeiro caso, serve para indicar ao serviço qual o objecto digital a ser comentado e quem o comentou. O ficheiro a ser enviado juntamente com o método *POST* terá o seguinte formato:

```
<Annotations>
  <Annotation pid="empid:1" username="cpatricia">
  Preciso de um ficheiro que contenha dados sobre a Dengue
  </Annotation>
</Annotations>
```

Neste caso dizemos que o objecto digital com *PID* empid:1 tem o comentário ”Preciso de um ficheiro que contenha dados sobre a Dengue”e quem o comentou foi o *username* cpatricia. Caso o utilizador sinta necessidade em comentar vários objectos digitais apenas precisa de adicionar ao *XML* a *tag Annotation* com o mesmo formato ao que está aqui apresentado.

Em relação ao método *PUT* requer outro tipo de ficheiro *XML*. Neste caso o que se pretende é alterar o estado de *like* de um comentário. Vamos supor que um utilizador concorda com um comentário apresentado no seu objecto digital e para isso pode invocar o serviço *web Annotation* com o seguinte ficheiro de entrada:

```
<Annotations>
  <Annotation pid="empid:179" id="23"/>
</Annotations>
```

Aqui está a informar o sistema que pretende colocar *like* no comentário 23 do objecto digital com *PID* empid:179.

Sistema de Pedidos - *request*

Este é sem dúvida um sistema muito interessante, uma vez que tira partido de uma funcionalidade do *Fedora Commons*. Como já se percebeu tirou-se o máximo partido do *Fedora Commons*. Desta vez prendeu-se com o facto de todos os *requests*, que estão dentro de uma colecção, não poderem aparecer aquando da visualização das mesmas no repositório. Para informar o repositório que irá conter uma colecção mas que esta não pode ser mostrada ao utilizador, foi preciso apenas especificar na colecção, que vai conter todos os *requests*, que o seu estado estará *inactive* em relação a todos os outros objectos digitais. E para realizar esta operação apenas foi preciso invocar uma função do *Fedora Commons*, nomeadamente *modifyObject* e colocar a *flag state* a *Inactive*. Assim temos uma colecção especial, onde é impossível ao utilizador visualizá-la. Todos os *requests* irão para essa colecção especial e ”ganham”automáticamente a sua característica, ou seja, são invisíveis. Caso o utilizador queira perceber quais os *requests* existentes no repositório terá que utilizar um *URL* específico com o método apropriado.

Método	URL	Descrição
GET	http://api.epimarketplace.net/request/list	Permite a visualização de todos os *requests* do repositório
POST	http://api.epimarketplace.net/request	Permite adicionar novos *requests* ao repositório
PUT	http://api.epimarketplace.net/request	Permite alterar o estado do *request*, ou seja o *Like*

Tabela 4.3: Métodos utilizados para a utilização do serviço *web request*

A tabela 4.3 ajuda a perceber como podemos utilizar este sistema de pedidos:

É preciso ter atenção ao métodos *POST* e *PUT*.

Para se colocar um *request* no repositório terá que se utilizar o método *POST* com um ficheiro de entrada *XML* específico. Neste caso, é preciso indicar qual o *username* que pretende inserir o *request*, a data desse mesmo pedido, o título, um *subject* e a própria mensagem em si.

Como um exemplo, pode visualizar em baixo:

```
<Request username="fjmc" date="2012-01-17">
  <title>
   Data describing the population's infection network structure
  </title>
  <type>Epidemiological data</type>
  <subject>Epidemiology</subject>
  <request>
   Data describing the population's infection network structure
   and collected at the beginning of the H1N1 pandemic outbreak
   in Israel in the summer of 2009 used in "Modelling the initial
   phase of an epidemic using incidence and infection network
   data: 2009 H1N1 pandemic in Israel as a case study",
   L. Stone et al., 2011.
  </request>
  <likes/>
</Request>
```

À semelhança do serviço *web listAnnotation*, o serviço *request* também permite colocar um *like* no pedido. O objectivo desta funcionalidade é a mesma em relação ao *Like* do *Facebook*. O ficheiro de entrada, no entanto, é diferente, basta ter o seguinte:

```
<Requests>
  <likes>
    <like username="cpatricia"></like>
```

```
  </likes>
</Requests>
```

Criar novas colecções no repositório - *category*

Este serviço *web* é sem dúvida o cerne da construção de toda a estrutura do repositório. Com ele é possível adicionar novas colecções ao repositório e definir onde se pretende colocar essas collecções.

Este serviço *web* utiliza o método *POST* com a *URL* `http://api.epimarketplace.net/category` e um ficheiro de entrada. Este ficheiro de entrada é muito simples e pode ser visualizado em baixo,

```
<Categorys>
  <Category username="fedoraAdmin" date="2011/11/10">  [1]
    <title>emresources</title>  [2]
    <category>emresources</category>  [3]
    <parentPid>root</parentPid>  [4]
    <description>Root collection</description> [5]
  </Category>
</Categorys>
```

A parte interessante do serviço *web* é permitir a inserção de várias colecções ao mesmo tempo num único ficheiro, neste caso, categorias e para isso é só adicionar mais *tags category* ao ficheiro *XML*. No exemplo em cima é inserido apenas uma colecção. Na linha (1) é informado quem é que vai criar a colecção e a data dessa inserção. Nas linhas seguintes é possível especificar com mais detalhe o título da colecção (2), a categoria dessa colecção(3), o caminho onde deve estar a colecção (4) e um breve descrição (5) para se perceber do que se trata a colecção.

Como o serviço *web* valida o caminho descrito em (4) é simples, verifica para cada *PID* se este é ou não colecção. Caso queiramos referir que o objecto, que queremos inserir, se encontra no objecto digital empid:1 então colocamos na *tag parentPid* o seguinte,

```
<parentPid>root/empid:1</parentPid>
```

Após todas as validações prossegue-se à criação do metadado *EM* com a pouca informação que o ficheiro *XML* oferece. A imagem 4.19 mostra o conteúdo do objecto digital, sendo este uma colecção.

Como se pode verificar a informação é reduzida mas é o suficiente para se perceber que se trata de uma colecção principal. A numeração contida na imagem serve para se

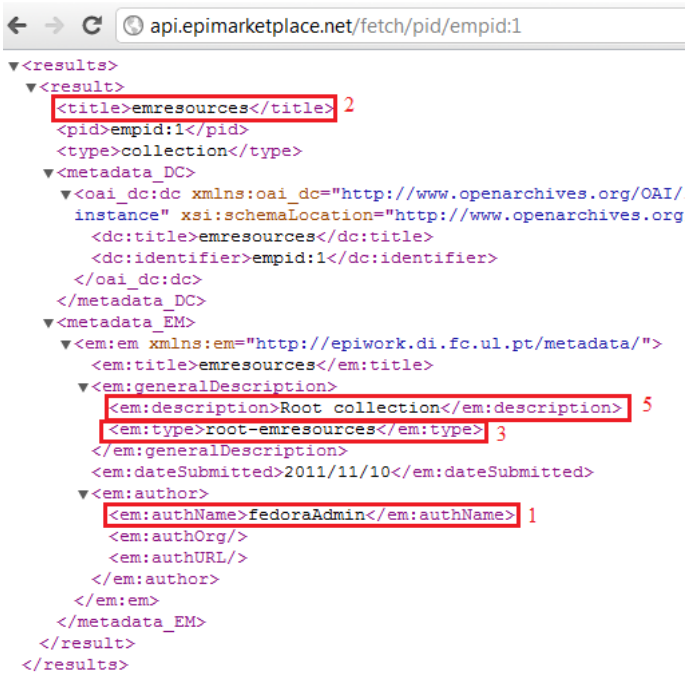

Figura 4.19: Conteúdo de um objecto digital que é categoria

perceber onde é colocada a informação proveniente do *XML*. O que pode suscitar dúvidas será a *tag* relacionada à numeração (3). A estrutura do repositório está definida como tendo uma base, pode-se chamar de base zero, onde é dito ser o nível *root*. Aqui é o ponto de partida para a criação de novas colecções. Caso queiramos criar uma nova colecção então esta irá estar dentro da ”base zero”, ou seja *root*, esta irá ter um caminho específico para ser acedida posteriormente, como num sistema de ficheiros *Linux* ou *Windows*. Assim podemos dizer que a nova colecção está em *root* e vai ter o *PID* empid:1, por exemplo, desta forma o caminho desta pasta será root-empid:1.

4.1.3 *Sistema de Gestão de Conteúdos*

Em relação à componente SGC, Sistema de Gestão de Conteúdos é responsável por apresentar uma *interface* gráfica ao utilizador. O Sistema de Gestão de Conteúdos utilizado no *Epidemic Marketplace* é o *Drupal* v6.17 e pode ser visualizado na seguinte *URL* `http://v2.epimarketplace.net` ou `http://www.epimarketplace.net`. A figura 4.20 apresenta o *layout* da página.

O *Drupal* permite fazer a ligação ao repositório de uma forma simples e eficaz. Este liga-se ao repositório através dos Serviços *web*.

É utilizado um módulo para fazer a gestão de autenticação dos utilizadores, *ldap_integration*, com este é possivel ligar-se ao *OpenLDAP* e obter toda a informação do utilizador.

Por forma a visualizar a utilização de alguns dos serviços *web* num ambiente gráfico,

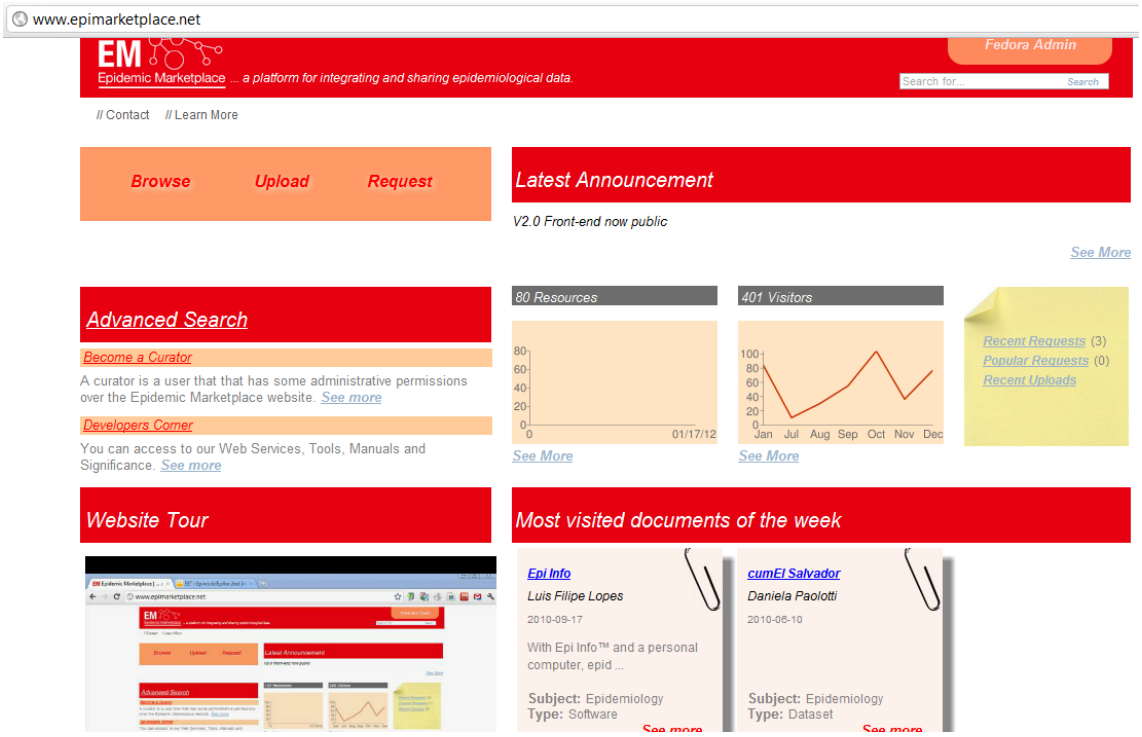

Figura 4.20: Página principal do *Epidemic Marketplace*

serão dadas a seguir *subsecções* com imagens que ilustram essas acções.

Todo o ambiente gráfico foi desenvolvido por mim à excepção do *upload*. No entanto, desde Agosto de 2011 que deixei de pertencer ao grupo *XLDB* levando a que a equipa do projecto *Epiwork* realizasse alguns ajustes no *layout*. Todas as imagens aqui mostradas apresentam o *layout* do *EM* actual. Segue-se uma breve descrição dessas imagens.

Serviço Web de Pesquisa sobre os dados do repositório - *search*

O exemplo da figura 4.21 trata-se de uma pesquisa sobre *subjects* que contêm a palavra *Epidemiology*. Neste caso pode visualizar-se que a pesquisa realizada no *Drupal* (do lado esquerdo da imagem) é igual à pesquisa realizada no Serviço *Web, search* (lado direito).

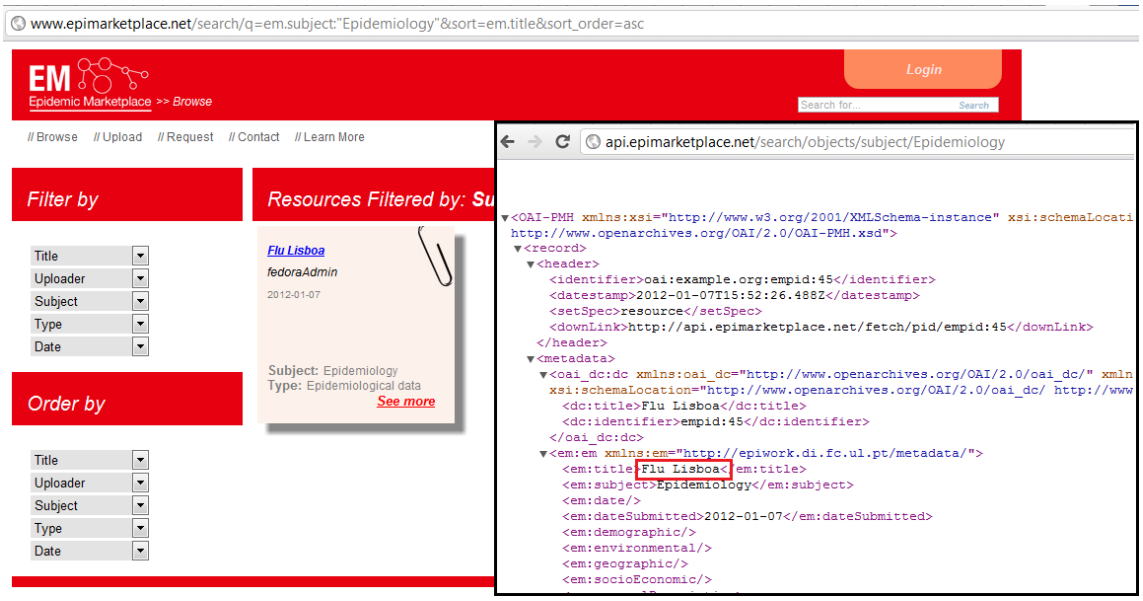

Figura 4.21: Pesquisa realizada no *Drupal*

Download de um objecto digital - *Fetch*

O que se pretende é visualizar todo o conteúdo de um objecto digital e para isso foi escolhido o empid:110 (figura 4.22). Mais uma vez do lado esquerdo encontra-se o exemplo no ambiente *Drupal* e do lado direito o exemplo com o Serviço *Web*, *Fetch*.

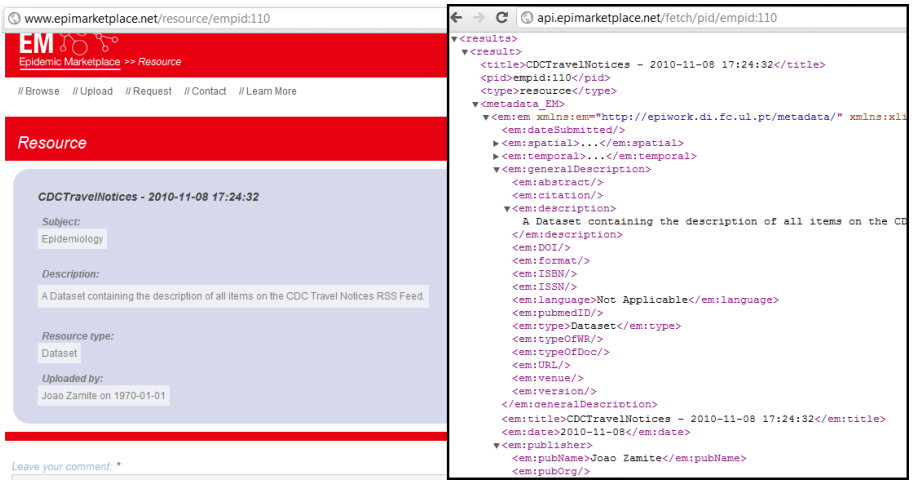

Figura 4.22: Visualização de um objecto digital no *Drupal*

Apresentar a estrutura do respositório - *repositoryTree*

É apresentado todo o conteúdo contido no objecto empid:1, figura 4.23. Aqui é muito mais perceptível para o utilizador o modo como estão estruturados todos os objectos digitais. Neste caso é semelhante a um sistema de ficheiros do *Linux* ou *Windows*.

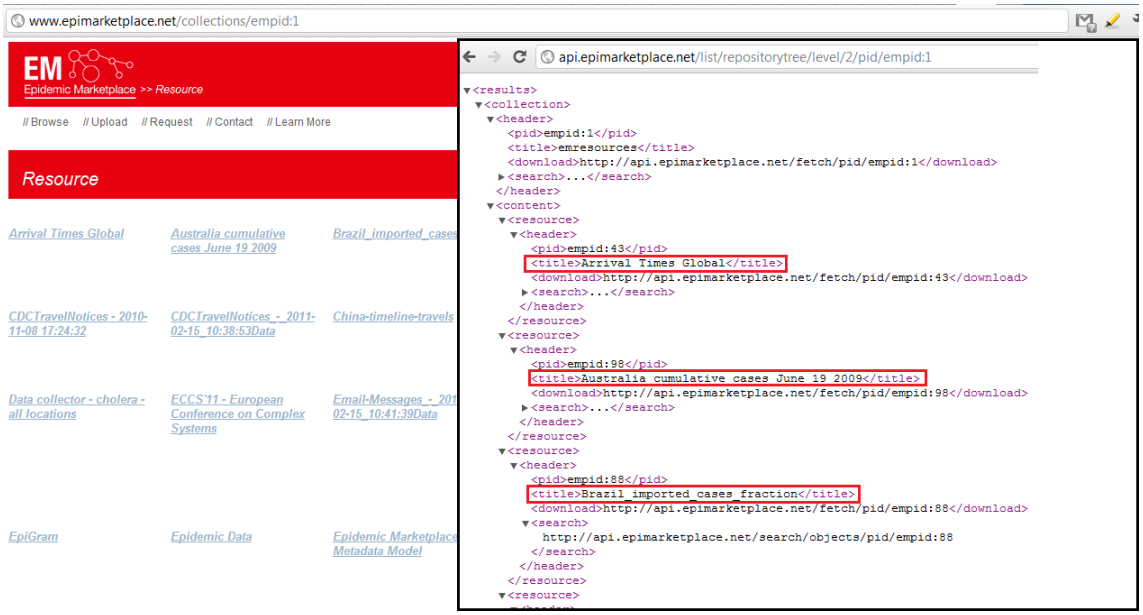

Figura 4.23: Visualização da estrutura do repositório no *Drupal*

Visualização de todos os comentários realizados no sistema - *listAnnotations*

Para se poder comentar um dado objecto digital, é necessário primeiro visualizar esse objecto e no fim da página encontra-se um espaço dedicado aos comentários, como mostra a figura 4.24. Neste caso quem comentou o objecto digital empid:3 foi o utilizador com *username* Zamite.

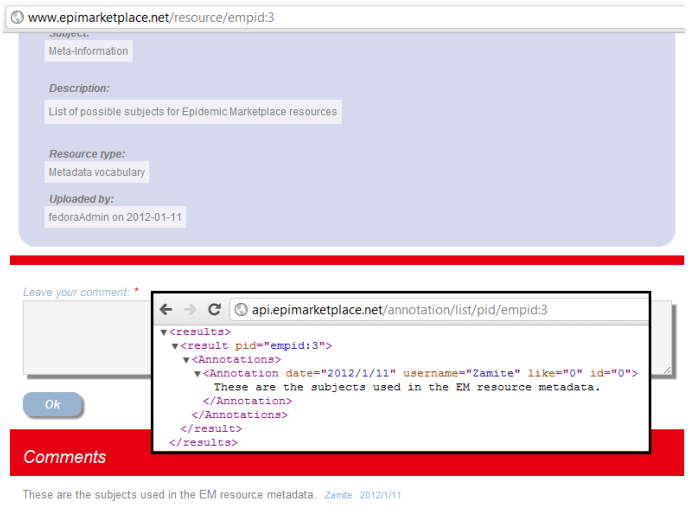

Figura 4.24: Comentar um dado objecto digital no *Drupal*

Sistema de Pedidos - *request*

Na figura 4.25 é visualizado um pedido que foi feito ao *Epidemic Marketplace* pelo utilizador com *username* fjmc.

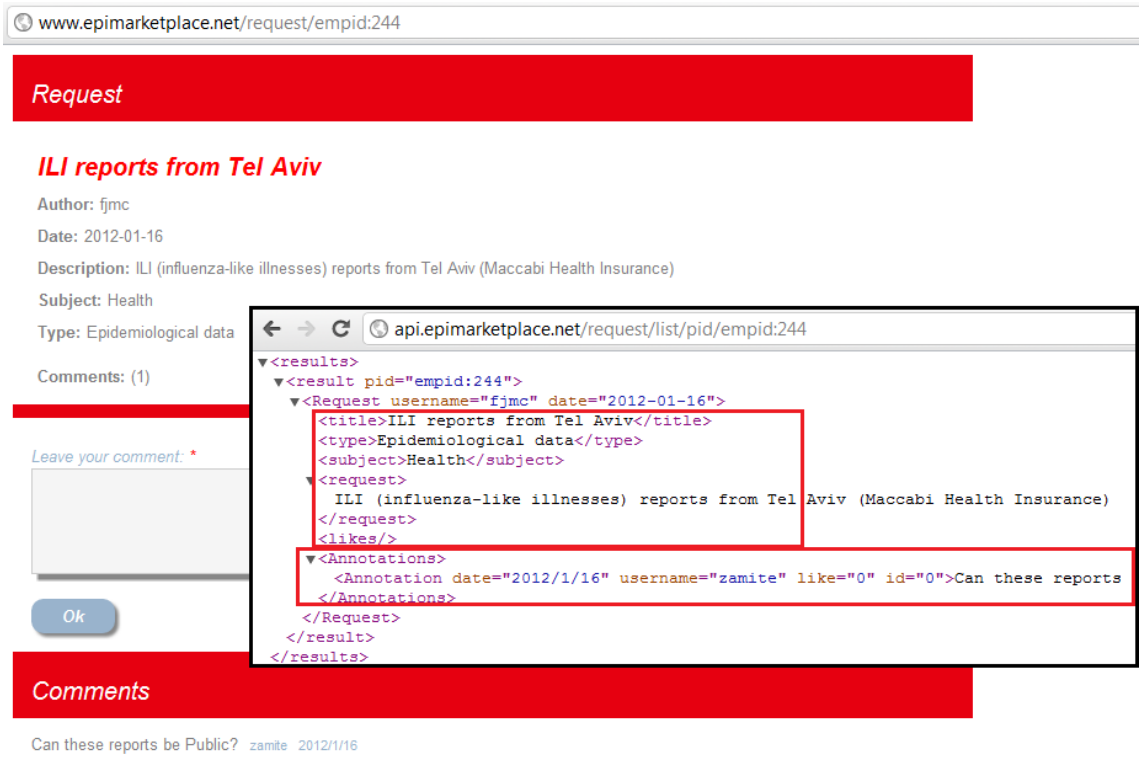

Figura 4.25: Visualização de um pedido realizado no *Drupal*

Criar novas colecções no repositório - *category*

Nesta secção é possível verificar como se insere uma colecção, categoria, no repositório. Do lado esquerdo da figura 4.26 encontra-se esta funcionalidade em ambiente *Drupal* e do lado direito da mesma o ficheiro que é enviado ao serviço *web category*.

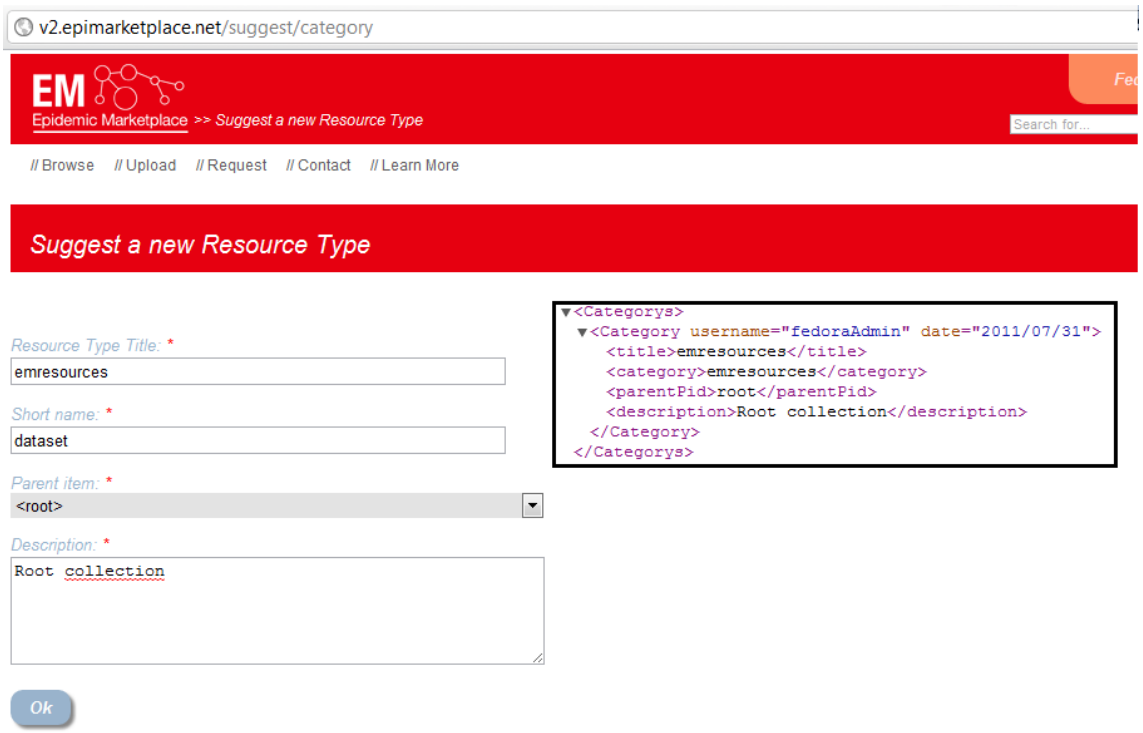

Figura 4.26: Inserção de uma colecção em ambiente *Drupal*

4.2 *Deployment*

Esta secção visa mostrar como se constrói todo o sistema, após se perceber qual o *software* utilizado é possível visualizar as diferentes camadas na figura 4.27.

Existem duas configurações activas, a de desenvolvimento e a de produção. A configuração de produção inclui o *software* aplicacional desenvolvido para o *Epidemic Marketplace* 1.0 e os seus componentes de *middleware*, enquanto que a configuração de desenvolvimento tem o *software* do *Epidemic Marketplace* 2.0. A versão de produção está acessivel por *Internet* em [20] e suporta quer os desenvolvedores quer a comunidade de epidemiologistas do projecto *Epiwork*, enquanto que a versão de desenvolvimento está apenas acessivel pela equipa do *Epiwork* (FCUL).

O projecto Epiwork tem as seguintes camadas:

- Camada de Hardware: São usados dois servidores fisicos, *Dell PowerEdge*. Ambos têm 8 CPU, 15 GB de memória e cada um com 2TB de disco.

- Camada de virtualização Xen: Os servidores *Dell* tem uma camada de virtualização instalada, através da plataforma open source Xen [32] e foram criadas 7 máquinas virtuais. *Xen*, permite que vários sistemas operacionais sejam acomodados de forma

heterogénia e executados no mesmo computador de forma concorrente. 9 das 7 máquinas virtuais tem 2 *CPU* virtuais e 1 GB de memória. As outras 2 máquinas virtuais tem 4 CPU virtuais e 2GB de memória. As 7 máquinas virtuais tem 10GB de disco e foi implementado um protocolo de rede para sistemas de ficheiros (NFS) [19].

Como foi dito em cima, existem duas configurações activas, a de desenvolvimento e a de produção. Para a configuração de produção, são usadas 3 máquinas virtuais com algumas diferenças na sua especificação. O sistema *Epidemic Marketplace* necessita de muita memória e CPU porque este corre sobre *Java Server Pages (JSP)*, desta forma foi escolhida a máquina virtual com 4 CPU, 2GB de memória e 10GB de disco. As outras 2 máquinas virtuais tem a segunda versão do *Epidemic Marketplace*, estas máquinas são constituidas por 2 CPU virtuais, 1 GB de memória e 10GB de disco.

A configuração de desenvolvimento usa as restantes máquinas virtuais.

- Camada de Sistema Operativo: Cada máquina virtual tem o sistema operativo CentOS instalado.

- Camada de Middleware: Na figura abaixo (4.27) foi feita a distinção entre os pacotes do *Epidemic Marketplace* 1.0 e o *Epidemic Marketplace* 2.0.

  A escolha dos módulos usados nas diferentes versões do Epidemic Marketpace verifica-se na versão do *Fedora Commons*. O *Epidemic Marketplace* 1.0 tem o Muradora como o front-end e o *Fedora Commons* 2.2 como o back-end. O *Epidemic Marketplace* 2.0, usa o *Drupal* e algumas extensões dele como front-end e o *Fedora Commons* 3.4.1 como back-end.

  O LDAP contém os dados relativos aos utilizadores e a base de dados contém todo o conteúdo do *Drupal*.

- Camada aplicacional: O *Epidemic Marketplace* 2.0 irá substituir a versão da plataforma com o melhoramento da comunicação, organização e integração dos dados.

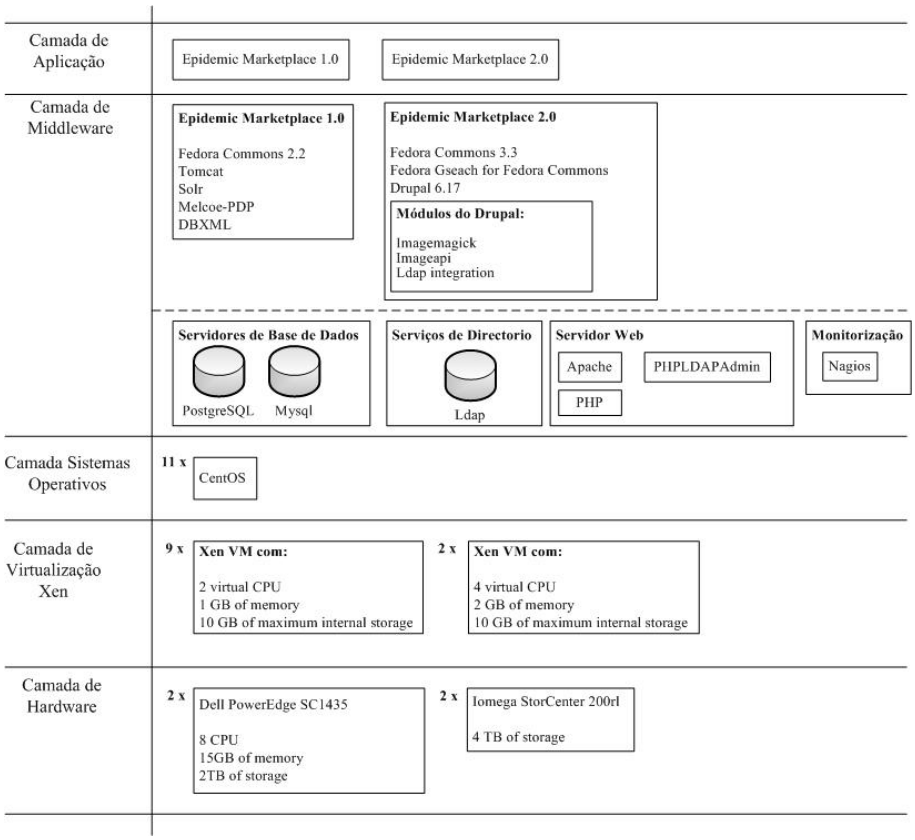

Figura 4.27: Estrutura em camadas de todo o sistema

Capítulo 5

Conclusão

O meu projecto foi desenvolvido no âmbito do Projecto *Epiwork* e visou melhorar o acesso e partilha dos dados existentes na plataforma *Epidemic Marketplace*. No primeiro ano de existência do *EM* foi realizada a concepção de uma plataforma e *web services*. Contudo, dada a dimensão dos dados e as novas necessidades, surgiu a necessidade de implementar uma nova versão do repositório e dos seus *web services* de modo a facilitar e agilizar o acesso e partilha dos dados.

É neste contexto que o meu projecto se desenvolve. Este foi dividido em duas tarefas principais:

- concepção de uma nova plataforma;
- realização de novas funcionalidades e alteração nos *web services* existentes.

O desenvolvimento da nova plataforma envolveu uma pesquisa profunda com base nos novos requisitos impostos ao *Epidemic Marketplace*. Os dados epidemiológicos estão guardados no *software Fedora Commons* v3.4.1 e o ambiente gráfico usufrui do ambiente Drupal e os seus respectivos módulos.

Os *web services* foram alvo de uma grande alteração, dado ser um requisito principal da plataforma, a integração com o novo modelo de metadados do repositório. O que fez com que todos os antigos serviços *web* deixassem de funcionar, dado que os *web services* estavam focados apenas para o padrão de metadados *Dublin Core*.

No âmbito geral, a nova versão do *Epidemic Marketplace* apresenta uma melhoria a nível de agilidade, comunicação, organização e integração dos dados.

Esta nova versão permite ter um melhor controlo de acesso aos dados do repositório. As políticas *XACML* estão definidas nos próprios objectos digitais. Desta forma é possível definir com mais detalhe o grau de precisão com que se pretende dar aos mesmos, por exemplo, dar acessos específicos a nível do *datastream*.

A nova versão conta também com a entrada de dois novos sistemas, um de comentários a objectos digitais e um outro de pedidos à comunidade do *Epiwork*. Foi interessante a

forma de implementação destes dois sistemas, uma vez que foram utilizadas todas as funcionalidades que o *Fedora Commons* oferece.

A pesquisa sobre os dados do repositório também sofreu grandes alterações e uma delas foi a possíbilidade de o utilizador poder realizar pesquisas sobre o novo modelo de metadados, o *EM Metadata*. Este foi, sem dúvida, o grande desafio do meu trabalho.

Outra inovação desta nova versão foi a implementação do serviço *web upload*. A grande vantagem desse serviço é o facto de permitir vários *uploads* de metadados *EM* em diferentes colecções num único pedido ao servidor. Permite também adicionar vários ficheiros no repositório num único pedido ao servidor.

O serviço *web fetch* também sofreu alterações, sendo agora possível visualizar o objecto de uma forma fácil e perceptível mesmo para utilizadores não muito conhecedores de informática.

Como projecto futuro, penso ser pertinente realizar este trabalho para aplicações móveis. Hoje em dia, dada a portabilidade dos meios de comunicação, caso o utilizador tenha a plataforma, por exemplo no seu telemóvel, poderá recolher dados epidemiológicos em tempo real e enviá-los para o repositório.

Outro ponto que poderia ser abordado futuramente seria colocar um controlo de acesso no *Solr*. Neste momento, as pesquisas realizadas aos diferentes objectos não estão a ser validadas pelo *Solr*, este apenas efectua a pesquisa e retorna os dados correspondentes. Deste modo, o *Fedora Commons* fica sobrecarregado, uma vez que este tem que juntar aos resultados dados pelo *Solr*, os papéis dos utilizadores e validar para cada um se o utilizador que efectuou a pesquisa tem ou não permissão.

Há ainda a possibilidade da ocorrência de uma ”infiltração”entre o *Solr* e o *Fedora Commons*. Ou seja, acontecer que uma aplicação maliciosa se coloque entre os dois e obtenha o resultado da pesquisa, altere os dados e os devolva ao utilizador. Penso ser um ponto muito importante a ser revisto num futuro próximo.

Bibliografia

[1] Apache Solr. http://lucene.apache.org/solr. Acedido em 2011-04-14.

[2] Biblioteca Nacional da Finlandia. http://www.nationallibrary.fi/libraries/doria.html. Acedido em 2011-07-10.

[3] Biblioteca Nacional da Finlândia. http://www.tlrp.org/dspace/index.jsp.

[4] The Data Collection of Epidemic Marketplace in EPIWORK. http://www.epimarketplace.net/documentation/documentation.html. Acedido em 2011-07-13.

[5] Domain Name System. http://en.wikipedia.org/wiki/Domain_Name_System. Acedido em 2011-01-10.

[6] Drupal Home page. http://drupal.org. Acedido em 2010-10-28.

[7] Dspace Aplicação. . Acedido em 2011-08-09.

[8] DSpace Home page. http://www.duraspace.org. Acedido em 2010-10-20.

[9] Dspace User Support Manager. http://www.dspace.org/images/Training_Materials/dspaceusers1107.ppt. Acedido em 2011-08-19.

[10] Dublin Core. http://dublincore.org. Acedido em 2011-08-09.

[11] eCrystals – Southampton. http://ecrystals.chem.soton.ac.uk. Acedido em 2011-07-10.

[12] EM Metadata. http://www.epimarketplace.net/metadata/XML-metadata-schema.xml.

[13] The Epidemic Marketplace in EPIWORK. http://epimarketplace.net. Acedido em 2010-09-3.

[14] EPrints Home page. http://www.eprints.org. Acedido em 2010-10-5.

[15] The European EPIWORK project Home page. http://www.epiwork.eu. Acedido em 2010-09-1.

[16] eXtensible Access Control Markup Language. http://xml.coverpages.org/xacml.html. Acedido em 2011-06-11.

[17] Fedora Commons Home page. http://fedora-commons.org. Acedido em 2010-10-7.

[18] Interactive Tucana Query Language. http://docs.mulgara.org/tutorial/itql.html. Acedido em 2011-04-12.

[19] Network File System Protocol Specification. http://tools.ietf.org/html/rfc1094. Acedido em 2010-10-20.

[20] O Epidemic Marketplace. http://v2.epimarketplace.net. Acedido em 2010-09-3.

[21] O Mediador do Epidemic Marketplace Master Thesis. http://xldb.fc.ul.pt/xldb/publications/Ferreira:OMediadorDo:2011_document.pdf.

[22] Open Archives Initiative Protocol for Metadata Harvesting. http://www.openarchives.org/pmh. Acedido em 2011-08-02.

[23] OpenLDAP Home page. http://www.openldap.org/. Acedido em 2010-10-7.

[24] Página da Wikipédia - Metadados. http://pt.wikipedia.org/wiki/Metadados. Acedido em 2011-07-12.

[25] Pode ser acedido através de. http://www.jisc.ac.uk/publications/briefingpapers/2006/pub_digipreservationbp.aspx. Acedido em 2011-07-11.

[26] RDF/XML Syntax Specification. http://www.w3.org/TR/REC-rdf-syntax. Acedido em 2011-07-29.

[27] Rutgers University Libraries. http://rucore.libraries.rutgers.edu. Acedido em 2011-03-24.

[28] Sistema Operativo Fedora. http://fedoraproject.org/pt/. Acedido em 2011-07-19.

[29] Web Service Upload. http://v2.epimarketplace.net/upload_object.

[30] WordPress. `http://wordpress.com`. Acedido em 2011-02-10.

[31] The Wweb Services of Epidemic Marketplace in EPIWORK. `http://v2.epimarketplace.net/developers_corner`. Acedido em 2011-07-18.

[32] Xen Home page. `http://www.xen.org`. Acedido em 2010-09-1.

[33] Tim Berners-Lee. Uniform Resource Identifier (URI): Generic Syntax. *The Cornell/CNRI Experiments*, 2005.

[34] Catarina Botelho. A gestão da mudança na introdução de novos Sistemas de Informação. *O Desafio da Mudança*, Outubro 2003.

[35] Russell D. e Gangeni O. Computer Security Basics. *O'Reilley Associates Inc.*, 1991.

[36] Roy Thomas Fielding. Architectural Styles and the Design of Network-based Software Architectures. *Doutoramento em Informação e Ciência da Computação - Universidade da California*, 2000.

[37] Hélio Kuramoto. Software livre para bibliotecas digitais. *Brasília*, 2005.

[38] Marilia LEVACOV. Bibliotecas virtuais: (r)evolução? *Ci. Inf.*, 26(2), 1997. Brasília.

[39] Paulino Michellazo. Ferramentas de CMS parceiras de sucesso. *PHP Magazine*, Janeiro 2007.

[40] Simple Object Access Protocol. Simple Object Access Protocol.

[41] H. L. Finstei C. E. Youman R. S. Sandhu, E. J. Coyne. Role Based Access Control Models. *IEEE Comput*, 1996.

[42] C. Lagoze E. Overly S. Payette, C. Blanchi. Interoperability for Digital Objects and Repositories. *The Cornell/CNRI Experiments*, Maio 1999.

[43] Doug L. Simpson. Content for one: developing a personal content management system. *ACM*, 2005. New York.

[44] Representational State Transfer. Representational State Transfer.

[45] Marinho R. Teixeira S., Silva L. Tecnologias Open Source na Criação de Bibliotecas digitais. 2004.

[46] Arms W. Digital Libraries. *MIT*, 2000.

[47] World Wide Web Consortium (W3C). About the world wide web consortium (w3c).

[48] Mário J. Silva; Francisco M. Couto; Dulce Domingos; Juliana Duque; Hugo Ferreira; Luís F. Lopes; Daniela Paolotti; Fabrício Silva; Patrícia Sousa; João Zamite. D3.3 Public Release of the Epidemic Marketplace Platform. 2010.